Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nino Zalašček

Varna komunikacija med prenosnimi napravami z operacijskim sistemom Android

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Operacijski sistem Android je najbolj razširjen, a hkrati ne uživa najvišje možne stopnje zaupanja glede varnosti. Vsebine zelo kritične narave je zato precej tvegano izmenjavati med napravami in pri tem uporabljati zgolj storitve in aplikacije podjetja Google.

Zasnujte in razvijte prototip sistema in aplikacije za varno komunikacijo med prenosnimi napravami z operacijskim sistemom Android. Sistem naj omogoča varno pošiljanje in sprejemanje tekstovnih in večpredstavnostnih vsebin, kot so: video, slika, zvočni posnetek. Za prenos naj se uporabljajo XML paketi. Poleg pošiljanja naj sistem omogoča tudi varno shranjevanje vsebin, ki so shranjene v šifrirani obliki. V primeru nedosegljivih naprav naj sistem vsebino zadrži in jo dostavi, ko je naprava dosegljiva.

Zahvaljujem se mentorju doc. dr. Roku Rupniku za strokovno vodenje pri izdelavi diplomske naloge. Zahvaljujem se tudi Anji za pomoč in podporo pri izdelavi diplomskega dela.

Diplomsko delo posvečam staršem Danili in Marijanu za vse spodbudne besede, podporo in pomoč v času študija.

Kazalo

Kazalo programskih kod

Slike

Seznam uporabljenih kratic

kratica	angleško	slovensko
AES	advanced encryption standard	napreden enkripcijski standard
RSA	rivest shamir adleman	rivest shamir adleman
MAC	message authentication code	koda za preverjanje pristnosti sporočila
HMAC	keyed-hash message authentication code	koda za preverjanje pristnosti sporočila z razpršilno funkcijo
XML	extensible markup language	razširljiv označevalni jezik
JSON	javascript object notation	objekt javascript notacije
XMPP	extensible messaging presence protocol	razširljiv sporočevalni prisotnostni protokol

Povzetek

Naslov: Varna komunikacija med prenosnimi napravami z operacijskim sistemom Android

Avtor: Nino Zalašček

V sklopu diplomskega dela smo razvili prototip aplikacijo za varno komunikacijo med prenosnimi napravami z operacijskim sistemom Android. Aplikacija omogoča varno pošiljanje in sprejemanje tekstovnih in večpredstavnostnih vsebin, kot so video, slika, zvočni posnetek. Za prenos se uporabljajo XML paketi. Tekstovne vsebine se prenašajo v paketu, za ostale vsebine pa paket vsebuje le identifikacijske številke, s katerimi aplikacija dostopa do dejanskih vsebin, ki so shranjene na zunanji podatkovni bazi in jih prikaže kot del sporočila. Aplikacija nudi poleg pošiljanja tudi varno shranjevanje vsebin, ki so shranjene v šifrirani obliki.

Komunikacija med napravami je posredna. Prenos sporočil poteka od naprave, kjer je bilo sporočilo ustvarjeno, do strežniškega sistema, ki sporočilo naprej pošlje do ustreznih naprav, ki so v sporočilu navedene kot prejemniki. V primeru nedosegljivih naprav, sistem sporočilo zadrži in jo dostavi, ko so naprave dosegljive.

Za prenos sporočil je uporabljena XMPP tehnologija, za dostop in shranjevanje podatkov v bazi pa so zadolžene HTTP zahteve. Za varnost skrbita simetrična kriptografija AES in asimetrična kriptografija RSA, za integriteto in avtentičnost sporočila pa skrbi HMAC. Za implementacijo vseh potrebnih tehnologij je uporabljen programski jezik Java.

Ključne besede: Android, Java, RSA, AES, HMAC, XMPP, varna komunikacija.

Abstract

Title: Safe communication across portable devices with Android operation system

Author: Nino Zalašček

Within thesis we developed an application prototype of the safe communication across portable devices with Android operation system. Application offers safe sending and receiving of text messages along with the other multimedia content like images, videos and sound recordings. XML packets are used for transferring data. Text content is stored and transferred inside the packet, for other multimedia content the packet contains the identification number, which is used by the application to access the actual content, which is stored on external database and present it as part of the message. Apart from sending and receiving, application also offers safe storing for all content, which is stored in encrypted form.

Communication between devices is indirect. Message is transferred from the device, where the message was created, to the server system which sends the message to the devices, which are listed in the message as receivers. In case of offline devices, system holds the message and delivers the message when devices are online.

XMPP technology is used for transferring messages and HTTP requests are used for accessing and storing data in the external database. For protection are used symmetric encryption AES and asymmetric encryption RSA. For the integrity and authentication is used HMAC. Java language is used for the implementation of all necessary technologies.

Keywords: Android, Java, RSA, AES, HMAC, XMPP, safe communication.

Poglavje 1

Uvod

1.1 Splošno o komunikacijski zasebnosti

Zasebnost je možnost posameznika ali skupine, da se obdrži zase oziroma obdrži informacije o sebi in tako izraža sebe oziroma informacije o sebi selektivno [3]. Meje in vsebina tega, kar smatramo za zasebno, se razlikujejo med različnimi kulturami in posamezniki, vendar glavne teme veljajo za večino. Kar posameznik smatra za zasebno, je zanj največkrat nekaj posebnega oziroma občutljivega. Definicija zasebnosti s pravnega vidika ni jasno definirana, zato se v primeru sodnega spora tehta pravica do zasebnosti naproti drugi pravici [5]. Poznamo štiri področja, na katera lahko razdelimo zasebnost. Prva je informacijska zasebnost, ki zavzema zbiranje in upravljanje z osebnimi podatki in jo poznamo tudi kot varovanje osebnih podatkov. Pri telesni zasebnosti gre za nedotakljivost osebnih delov človekovega telesa. Komunikacijska zasebnost je zasebnost vseh oblik sporazumevanja. Sledi še zadnja zasebnost, tj. zasebnost v prostoru, ki zavzema omejevanje poseganja v osebni prostor človeka [4].

Z zasebnostjo je povezan tudi nadzor. Nadzor je lahko dober ali slab. Danes je nadzorovanje posameznikov sredstvo družbenega nadzora, kakor tudi sredstvo za zagotavljanje pravic družbene participacije. Pri nadzoru ne gre mimo tehnologije. Informacijske tehnologije so namenjene zbiranju in

obdelavi vseh vrst podatkov in informacij. Lahko rečemo, da je informacijska družba družba nadzora. Informacijske tehnologije imajo danes izjemen pomen za državno varnost.

V današnjem času človek uporablja veliko tehnologije, ki za svoje delovanje uporablja tudi internet. To olajša naše življenje na nešteto načinov, kot so brskanje po internetu, dokumenti na oblaku, objavljanje slik na socialnih omrežjih, dostop do oddaljenih naprav in tako naprej. Vse to vključuje pošiljanje in prejemanje podatkov. Vsak podatek, ki je poslan ali prejet, ni več dostopen samo pošiljatelju in prejemniku, pač pa lahko do podatka dostopa še tretja oseba. Seveda je v primerih, ko se objavi fotografije na socialnem omrežju, namen tega dejanja omogočiti dostop tudi tretjim osebam. Ko pa se dve osebi pogovarjata v privatnem pogovoru, nočeta, da bi pogovor lahko videl ali slišal še kdo drug, a lahko oseba z ustreznim znanjem in orodjem to vidi in posluša. V teoriji lahko vsak lastnik omrežja interneta posluša oziroma vidi podatke, ki jih pošiljajo naprave preko njihovega omrežja. To je lahko ponudnik interneta, brezžično omrežje in tako naprej. Obstaja tudi možnost, da se nekdo priključi na omrežje in začne prisluškovati.

Večja kot je uporaba naprav povezanih z internetom, manjša je zasebnost. V teoriji je lahko vsak podatek, ki je na z internetom povezani napravi, na voljo drugim osebam. V praksi je zasebnost posameznika ali skupine velikokrat zlorabljena, vendar se posameznik ali skupina tega ne zaveda.

1.2 Vsebina diplomske naloge

Diplomska naloga opisuje katere tehnologije in orodja so potrebne za pošiljanje in prejemanje sporočil, kot tudi za šifriranje in dešifriranje teh sporočil. Kako tehnologije sovpadajo med seboj je vidno med opisom implementacije. Poleg tega so opisane tudi tehnologije in orodja, ki omogočajo lažje in boljšo pisanje izvorne kode.

Opisani so tudi postopki za izdelavo Android [13] aplikacije, od načrta do same implementacije razredov. Pri tem se jasno vidi, kako in katere

tehnologije so uporabljene pri določeni implementaciji. Prikazano je kako aplikacije povezuje XMPP [16] strežnik in zunanjo podatkovno bazo, da na videz delujeta kot eno. XMPP strežnik skrbi za takojšen prenos podatkov med napravami, zunanja podatkovna baza pa nudi shranjevanje podatkov, ki so za varno komunikacijo potrebni. Poseben poudarek je na kriptografiji. Opisano je kje in kako so določeni ključi shranjeni in pri katerih kriptografskih operacijah so uporabljeni.

Diplomska naloga bralcu ponuja podlago, na kateri se lahko izdela aplikacijo, ki je namenjena varnemu pošiljanju in prejemanju sporočil.

Poglavje 2

Zaščita podatkov

Z dneva v dan je na svetu več zasebnih podatkov in prav tako več zlorab. Za preprečevanje zlorab obstaja vedno več načinov za zaščito podatkov.

2.1 Digitalni vodni tisk

Svetovni splet vsebuje ogromne količine večpredstavnih vsebin: slik, grafik, videoposnetkov, tekstov in zvočnih zapisov. Te vsebine so na voljo vsem, ki imajo dostop do interneta. Ker lahko spletni uporabniki pregledujejo in prenašajo te vsebine, pride do težave z zaščito avtorskih pravic. Za tovrstno zaščito obstaja tehnika, ki se imenuje digitalni vodni tisk [7].

Zaščite zaupnih podatkov vključujejo šifriranje podatkov, digitalni vodni tisk pa zaščiti večpredstavne vsebine, ki so na voljo v nešifrirani obliki. Običajne oblike zaščite podatkov onemogočajo dostop do podatkov, digitalni vodni tisk pa se uporablja v procesu dokazovanja, ko je že prišlo do protizakonitega dejanja. Digitalni vodni tisk deluje tako, da je v primarni digitalni signal vstavljen sekundarni digitalni signal, ki ga je mogoče zaznati ali izločiti. Pri slikah je sekundarni signal slika vodnega znaka, ki je lahko logotip, sporočilo ali naključno ustvarjen vzorec ali slika. Vodni znak je odvisen od namena in vloge vodnega tiska. Vloge vodnega tiska so potrjevanje lastništva ter preprečevanje nepooblaščenega razmnoževanja in razširjanja,

preverjanje verodostojnosti in celovitosti, skrito označevanje vsebine, kontrola uporabe in oglaševalne vsebine. Poleg različnih vlog vodnega tiska poznamo več tehnik. Vodni tisk je lahko viden ali neviden in krhek ali robusten.

Vidni tisk vsebuje vidni vzorec, sporočilo ali logotip organizacije. Vodni tisk po kvaliteti določata dva kriterija: kako težko je vgrajeni vodni tisk odstraniti v primeru nepooblaščenega odstranjevanja in ali vgrajeni vodni tisk zagotavlja, da ga je vstavila oseba, na katero se sklicuje. Slika, ki vsebuje nevidni vodni tisk, je na videz enaka sliki, ki ne vsebuje vodnega tiska, ni pa identična. Prisotnost lahko v tem primeru odkrijemo le z ustreznim algoritmom.

Krhek vodni tisk lahko pokvarimo s katerokoli tehniko obdelave slike. Takšnega tipa je vodni tisk za ugotavljanje verodostojnosti in celovitosti vsebine. Robustni vodni tisk je odporen proti različnim oblikam ponarejevanja ali odstranjevanja in proti operacijam obdelave večpredstavnostne vsebine. Primeren je za potrjevanje lastništva. Kljub vsem tehnikam robustnega tiska se še vedno najdejo načini za odstranjevanje in ponarejevanje vsebin. Pri izločanju vodnega tiska obstajata dva načina. Vodni tisk je lahko izločen v originalni obliki, pri nekaterih vsebinah pa lahko izvedemo le potrditev prisotnosti določenega vodnega tiska. Izločanje oziroma potrjevanje vodnega tiska običajno zahteva varnostni ključ. Sheme vodnega tiska, ki uporabljajo ključ, delimo na sheme tajnega ključa in sheme javnega ključa. Pri tajnem ključu se uporablja isti ključ pri vstavljanju in izločanju ali potrjevanju pristnosti vodnega tiska. Pri javnem ključu je ključ za vnos drugačen kot za izločanje ali potrjevanje pristnosti. Za vnos se uporablja tajni ključ, za izločanje ali potrjevanje pristnosti pa javni ključ. Posebne so še sheme, ki za izločanje ali potrjevanje potrebujejo originalno sliko. Taka shema se imenuje zasebna shema. Shema, ki originalne slike ne zahteva, se imenuje shema s pozabo.

Poleg digitalnega vodnega tiska se uporablja tudi tehnika kriptografije. Razlika med tehnikama je v tem, da podatki niso dostopni oziroma razumni vsem, tako kot so pri digitalnem vodnem tisku.

2.2 Kriptografija

Kriptografija [6] ali zakrivanje sporočil spada v kriptologijo, ki je veda o tajnosti, šifriranju, razkrivanju šifriranih podatkov in zakrivanju sporočil. Kryptos logos, iz katere izhaja beseda kripto, je grška besede ki pomeni skrita beseda. Osnovno sporočilo je poznano kot čistopis (cleartext, plaintext), zašifrirano pa šifropis ali tajnopis (kriptogram, ciphertext). Sporočilo postane kriptirano, ko ga po nekem algoritmu ali metodi z določenim ključem spremenimo v kriptirano obliko. Da lahko prejemnik sporočilo razume, se morata z avtorjem sporočila dogovoriti o ključu in algoritmu ali metodi.

Kriptografija je v uporabi že od časa pred našim štetjem. Skozi čas je bilo razvitih nešteto oblik skrivanja sporočil. Špartanci so na valj navili ozek trak in sporočilo napisali pravokotno na smer traku. Odvit trak je bil poslan naslovniku, naslovnik pa je moral imeti valj z enakim premerom.

Eno izmed najbolj znanih oblik je uporabljal Julij Cezar, ki je pošiljal sporočila, v katerih je vsako črko zamenjal s črko, ki je bila nekaj mest za njo. Metode, pri katerih se črka vedno preslika v isto črko, je zelo preprosta za dešifriranje. Poiskane so najbolj pogoste črke in so nadomeščene z najbolj pogostimi črkami v jeziku. Metodo zamenjave črk so nadomestile varnejše poli alfabetske substitucije. Pri poli alfabetski substituciji so znaki iz osnovne abecede kriptirani z dvema ključema. Lihi znaki so kriptirani po prvi zamaknjeni abecedi, sode pa po drugi zamaknjeni abecedi. Tako je možnost razkritja čistopisa precej zmanjšana. Primer uporabe poli alfabetske substitucije je enigma, ki so jo Nemci uporabljali med 2. svetovno vojno. Poli alfabetsko substitucijo je nadomestila transpozicija.

Najenostavnejši primer transpozicije je stolpčna transpozicija. Čistopis je v obliki stolpcev in kolon, tajnopis pa se tvori tako, da se prepišejo znaki po stolpcih. Zaporedje stolpcev predstavlja ključ. Algoritmi s takim ključem spadajo med simetrične algoritme.

Pri simetričnih algoritmih [8] je samo en zasebni ključ, s katerim se vsebino zašifrira in dešifrira. Šifriranje je običajno hitro, vendar problem predstavljata izmenjava ključev in število ključev, ker mora imeti vsak uporabnik

svoj ključ za vsakega dopisovalca. V izogib tem problemov so razvili asimetrične metode. Pri tej metodi obstajata dva ključa, zasebni in javni. Javni ključ je javen in z njim lahko vsak šifrira sporočilo, a sporočilo bo lahko dešifriral le tisti, ki ima zasebni ključ. Asimetrične metode [9] so računsko bolj zahtevne in zato tudi počasnejše kot simetrične.

Sporočila mora običajno prepotovati neko pot od avtorja do prejemnika. Na tej poti lahko kdo sporočilo spremeni. Poznan je tudi problem identifikacije lastnika javnega ključa. Za varnost je potrebna zagotovljenost za zaupnost, celovitost, overjanje, preprečevanje tajenja in kontrolo dostopa. Zato se je razvilo podpisovanje sporočil in overjanje javnih ključev. Potrdilo vsebuje poleg podatkov o ključu še čas nastanka, podatke o lastniku, rok veljavnosti in podobno. Za verodostojnost sporočila je dodan digitalni podpis. To je z zgoščevalno funkcijo izračunan fiksni povzetek vsebine.

Poglavje 3

Tehnologije in orodja

3.1 Kriptografija

3.1.1 Simetrična kriptografija AES

Simetrična enkripcija [8] je način kriptografije, kjer se uporablja isti ključ za šifriranje in dešifriranje podatkov. Najbolj razširjen algoritem za simetrično enkripcijo je AES, ki ga je z javnim natečajem izbral ameriški Narodni urad za standarde in tehnologijo. AES je napreden enkripcijski standard, ki je nasledil DES. Poznan je tudi pod imenom Rijndael. Gre za specifikacijo enkripcije elektronskih podatkov, ki sta ga razvila belgijska kriptografa Vincent Rijmen and Joan Daemen in nato objavila leta 1998. AES uporablja ključe dolžin 128, 192 ali 256 bitov.

Poleg ključa, ki ga uporabljamo za šifriranje in dešifriranje podatkov, pri simetrični kriptografiji obstaja še inicalizacijski vektor ali začetna spremenljivka. Vektor je vnos fiksne velikosti, ki se uporablja za randomizacijo. Randomizacija je ključna za kriptografske sheme za zagotavljanje semantične varnosti. Nonce ali po slovensko skovanka je arbitrarna številka, ki je uporabljena samo enkrat v kriptografski komunikaciji v duhu pomena besede. Zadnji del asimetrične kriptografije je kriptografska sol. Gre za naključno vrednost, ki je uporabljena kot dodaten parameter v enosmerni funkciji, ki

razprši geslo. Nova vrednost soli je generirana za vsako geslo.

Pri asimetrični kriptografiji poznamo tudi več načinov bločnih šifer . Način bločne šifre je metoda, ki pri šifriranju aplicira deterministično algoritem skupaj z asimetričnim ključem na blok podatkov.

Ključi v asimetrični kriptografiji morajo biti visoko entropijski, da so težje zlomljivi. Prav tako dolžina ključa poveča zaupnost. Za generiranje naključnih vrednosti je potrebno uporabiti varne generatorje naključnih vrednosti. Generiranje ključev je možno tudi s funkcijami za izpeljavo ključa iz gesla, ki generirajo visoko entropijske ključe iz psevdo-naključne funkcije, gesla in s številom iteracij [2].

3.1.2 Asimetrična enkripcija RSA

Pri asimetrični kriptografiji [9] se uporabljata dva ključa. Javni ključ se uporablja za šifriranje, zasebni ključ pa za dešifriranje podatkov. Ker sta pri asimetrični kriptografiji dva ključa, je razširjanje ključev veliko lažje, kot pri simetrični kriptografiji. Javni ključ lahko javno objavimo, saj se ga uporablja samo za šifriranje. Zasebni ključ ima v lasti tisti, ki bo šifrirano vsebino dešifriral. Ena izmed prvih praktičnih kriptografij se imenuje RSA ali Rivest-Shamir-Adleman. To so priimki razvijalcev Rona Rivesta, Adia Shamirja in Leonarda Adlemana, ki so leta 1977 objavili RSA algoritem. RSA je pogosto uporabljen za varen pretok podatkov. Algoritem bazira na dveh velikih praštevilih, s katerima se generira javni ključ. Zaradi velikih praštevil je RSA relativno počasen algoritem, zato je redko uporabljen za direktno šifriranje podatkov, večkrat pa se ga uporablja za šifriranje simetričnih ključev, s katerimi se nato šifrira in dešifrira podatke [2].

3.1.3 Koda za preverjanje pristosti sporočila z razpršilno funkcijo HMAC

V kriptografiji se za preverjanje pristnosti sporočila uporablja MAC [2], ki potrjuje, da je sporočilo prišlo od navedenega pošiljatelja. Overitvena koda

sporočila ščiti integriteto in avtentičnost podatkov v sporočilu. Posebna vrsta za preverjanje pristnosti sporočila, ki dodatno vsebuje še razpršitveno funkcijo in skrivni ključ, se imenuje HMAC [10].

3.1.4 Zapolnjevanje

Zapolnjevanje ali po angleško padding pri kriptografiji pomeni dopolnjevanje nepopolnih podatkovnih blokov. Šifrirani vrednosti se doda neko vrednost in tako je dolžina šifriranih podatkov neznana za zunanjega opazovalca. Zapolnjevanje poznamo pri simetrični in asimetrični kriptografiji.

3.2 Android

Android je odprtokodni operacijski sistem, ki je zgrajen na podlagi Linux jedra. Android je od leta najpopularnejši mobilni operacijski sistem, ki ga je razvilo podjetje Android Inc. Kasneje je to podjetje kupil Google. Sprva je bil Android namenjen pametnim telefonom in tablicam z ekranom na dotik, sedaj pa ga zasledimo tudi na drugih elektronskih napravah. Razvijanje aplikacij za Android je mogoče z razvijalskim orodjem Android Studio [12], ki je brezplačno. Razvijanje [11] poteka v visokonivojskem jeziku Java ali po novem tudi v jeziku Kotlin. Aplikacije so objavljene v oblaku, ki se imenuje Google Play.

3.3 XMPP

XMPP [16] je zbirka XML tehnologij, ki so namenjene takojšnjemu sporočanju ter izmenjavi podatkov o pristnosti. Omogoča tudi izmenjavo strukturiranih podatkov v skoraj realnem času. "X"predstavlja "extensible", in pomeni, da je XMPP razširljiv. Narejen je tako, da rase in je sprejemljiv za spremembe. "M"predstavlja "messaging"oziroma sporočanje; to je del XMPP-ja, ki je viden. XMPP je narejen, da pošilja vsa sporočila v realnem času, z uporabo učinkovitega "push"mehanizma. "Pštoji za "presence"oziroma prisotnost;

gre za obveščanje strežnika o uporabnikovem statusu, ali je pripravljen na sprejemanje sporočil ali ne. Zadnji "P"predstavlja "protocol"oziroma protokol. XMPP je protokol oziroma zbirka standardov, ki omogoča različnim sistemom, da komunicirajo med sabo. XMPP je zelo razširjen, vendar ni tako zelo oglaševan. Protokol nadzoruje XSF, to je ustanovitev za standardizacijo razširitev.

3.4 Java

Java [14] je računalniški programski jezik, ki je objektno orientiran, bazira na razredih in podpira sočasne operacije, kar pomeni, da se lahko več operacij izvaja vzporedno. Jezik je narejen tako, da ima čim manj implementacijskih odvisnosti. Namen je, da bi aplikacije, napisane v Javi, morale delovati na vseh platformah. Java se smatra za najbolj uporabljen programerski jezik. Jezik je pod imenom Oak razvil James Gosling leta 1991. Leta 1995 je bila objavljena različica Java 1.0. Jezik je bil razvit v stilu sintakse C/C++.

3.5 Gradle

Gradle [21] je odprtokodni avtomatiziran sistem za pripravo programskih paketov, ki je bil zgrajen na konceptu Apache Ant in Apache Maven. Od omenjenih dveh se razlikuje v tem, da ne uporablja XML form, ampak razvojni jezik, ki bazira na Groovy-ju. Pri pripravi se izvršijo naloge, ki narekajo, kaj bo paket vseboval in kako bo zapakiran. Za določitev reda izvršenih nalog se pri tem sistemu uporablja usmerjen acikličen graf. Gradle je bil razvit za grajenje programske opreme, ki vsebuje več projektov in lahko postane zelo velik.

3.6 XML

Extensible Markup Language [22] je angleški izraz, ki ga predstavlja kratica XML. To je jezik, ki omogoča format strukturnih podatkov za prenos med omrežji. Jezik je razširljiv, saj omogoča poljubna imena etiket in je prilagodljiv potrebam razvijalcev. Sestavljen je iz treh delov, prvi je podatkovni, ki hrani podatke, drugi je deklarativni, ki skrbi za ločevanje med etiketami, ter predstavitveni, s pomočjo katerega je izpis podatkov oblikovan. Zaradi preproste in pregledne zgradbe je najbolj uporaben za komunikacijo, vendar se ga uporablja tudi za grajenje grafičnih uporabniških vmesnikov.

3.7 JSON

Sintaksa za shranjevanje in izmenjevanje podatkov se imenuje JSON [24]. Gre za tekst napisan v JavaScript objektni notaciji, ki je lahek za branje za človeka in stroje. Vsebina datoteke je zbirka, sestavljena iz parov imen in vrednosti. Največkrat je uporabljen za izmenjavo podatkov med klientom in strežnikom, kjer so podatki lahko samo v tekstovni obliki. Objekti so pretvorjeni v JSON obliko na strani pošiljatelja in so nato na strani prejemnika pretvorjeni nazaj v objekt. JSON format je prvi določil Douglas Crockford, prve uporabe pa so se pojavile v brskalnikih v zgodnjih letih 21. stoletja.

3.8 HTTP zahteve

HTTP [27] je komunikacijski protokol med klienti in strežniki ter deluje na podlagi para zahteve in odziva. Klient je običajno brskalnik, aplikacija ali računalnik, ki gosti spletno stran, pa je običajno strežnik. Trenutna verzija protokola je HTTP/1.1, ki ima nekaj več lastnosti kot predhodnik 1.0, glavna izmed lastnosti je omogočanje trajne povezave. HTTP je sestavljen iz URL-ja, HTTP glagolov, HTTP glav, statusnih kod in formata sporočila. Najbolj uporabljen je glagol GET, ki pridobi vsebino. Pogosto uporabljeni so še POST, PUT in DELETE. POST je uporabljen za ustvarjanje nove vsebine,

PUT je uporabljen za dodajanje ali spreminjanje obstoječe vsebine, DELETE pa je uporabljen za odstranjevanje vsebine[26].

3.9 Testiranje in analiza kode

3.9.1 Testiranje modulov

Testiranje modulov je metoda za testiranje programske opreme, kjer posamezne enote izvorne kode primerjajo module programske opreme s kontrolnimi podatki, procedurami uporabe in operacijskimi procedurami. Na podlagi testnih rezultatov je vidno, ali je program primeren za uporabo. Testni moduli so kratki fragmenti kode, ki jih programerji napišejo v procesu razvijanje programske opreme [25].

3.9.2 Avtomatsko testiranje

Kar precejšen del izdelka v Jenkinsu predstavlja avtomatsko testiranje. Rezultat izdelka je odvisen od avtomatskih testov, ki se sprožijo avtomatsko in morajo zadovoljiti zahteve, ki so določene za posamezen test. Avtomatski testi poenostavijo ponavljajoča dela, kakor tudi opravijo teste, ki bi jih težko izvedli ročno. Pisanje avtomatskih testov vzame nekaj dodatnega časa, a se ta kljub temu, da se teste poganja po vsakem določenem sklopu sprememb, vedno povrne. Avtomatski testi so kritični za neprekinjeno dostavo izdanih izdelkov ter neprekinjeno testiranje [1].

3.9.3 Pokritost kode

Pri testiranju programske kode je poleg testnih rezultatov pomembno, koliko kode so pokrili opravljeni testi. Večja kot je pokritost kode, manjša je možnost, da koda vsebuje hrošče in druge nepravilnosti. Za izračune pokritosti kode se uporabljajo različni kriteriji, najbolj osnovna sta odstotek programskih funkcij in odstotek programskih stavkov, ki sta izvedena med izvedbo testov. Obstajata še dva kriterija; prvi je pokritost vej, ki preverja,

če je bila vsaka veja kontrolne strukture poklicana, drugi pa se imenuje pogojna pokritost, ki preverja, če je bil vsak pogojni izraz izveden kot pozitiven in negativen [29].

3.9.4 Statična analiza kode

Statična analiza kode je proces, s katerim je ovrednotena kakovost kode, ne da bi bila pri tem izvedena. V večini primerov se statična analiza izvede na programski kodi. Analiza preveri obnašanje posameznih izrazov, deklaracij in spremenljivk. Rezultat analize so stavki iz programske kode s komentarji, kaj je narobe s stavkom in kakšna je stopnja nevarnosti same napake [30].

3.10 Orodja

3.10.1 Sistem za upravljanje z izvorno kodo Git

Najbolj razširjen sistem, ki skrbi za kontrolo različic, je Git [23]. Pri projektih, kjer več ljudi razvija isti del programske kode, je sistem Git še posebej uporaben. Z njegovo pomočjo se lahko prosto preklaplja med različicami programske kode. Sistem pospeši razvoj programske opreme, saj lahko vsak, ki ima pravico do repozitorija, vsak trenutek naloži spremembo v repozitorij, kakor tudi prenese spremembe iz repozitorija. Razvijalec lahko gleda v zgodovino svojih sprememb in zgodovino sprememb drugih razvijalcev. Ko več razvijalcev deluje na isti programski kodi, pride do konfliktov, ki jih Git zazna in opozori razvijalca, v nekaterih primerih pa konflikte popravi sam. Poleg pospeševanja sistem nudi tudi neke vrste varnost pred izgubo podatkov, saj ima vsak razvijalec pri sebi naložen cel repozitorij. Če se strežnik z repozitorijem sesuje, ima eden od razvijalcev pri sebi zadnjo verzijo.

3.10.2 Jenkins

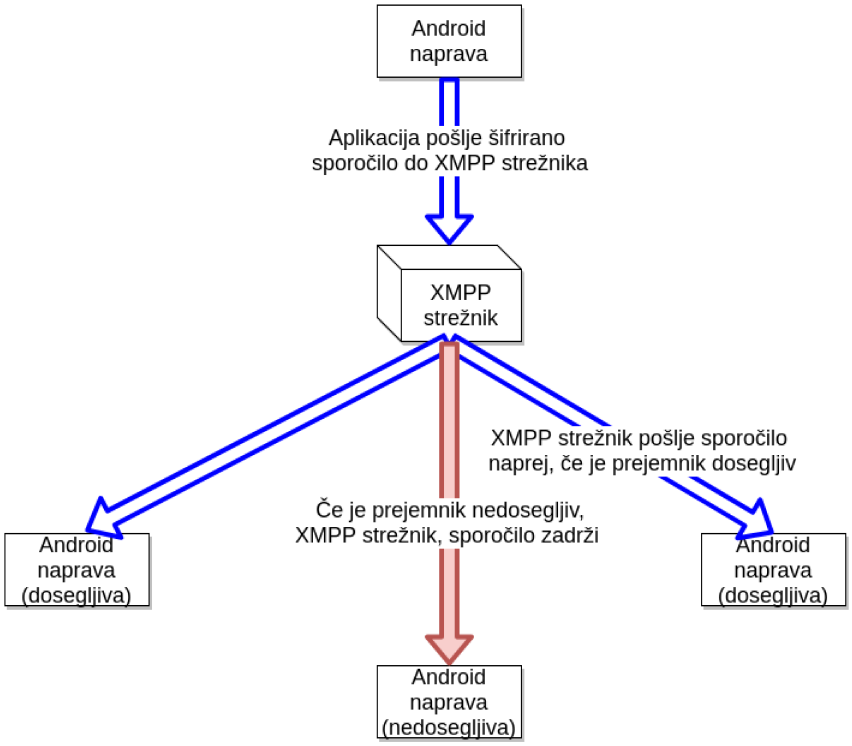
Odprtokodno orodje za avtomatizacijo procesov razvijanje programske opreme se imenuje Jenkins [28]. Razvil ga je Kohsuke Kawaguchi leta 2011. Jenkins

nudi neprekinjeno integracijo in olajšavo pri tehničnih oblikah neprekinjene dostave izdanih izdelkov. Je sistem, ki deluje na strežniku in podpira orodja za kontrolo različic, lahko poganja Apache in druga enostavna orodja skupaj z lupinskimi skriptami in drugimi ukazi. Funkcionalnost Jenkinsa je lahko razširjena z uporabo vtičnikov. Delo izdelka v Jenkinsu lahko sprožijo različni sprožilci, ki jih je mogoče nastaviti v nastavitvah vsakega izdelka ali pa se lahko nastavi različne cikle, ki sprožijo delo glede na nastavljen čas.

Poglavje 4

Zasnova aplikacije

4.1 Delovanje aplikacije na napravah



Slika 4.1: Prikaz delovanja aplikacije na napravah.

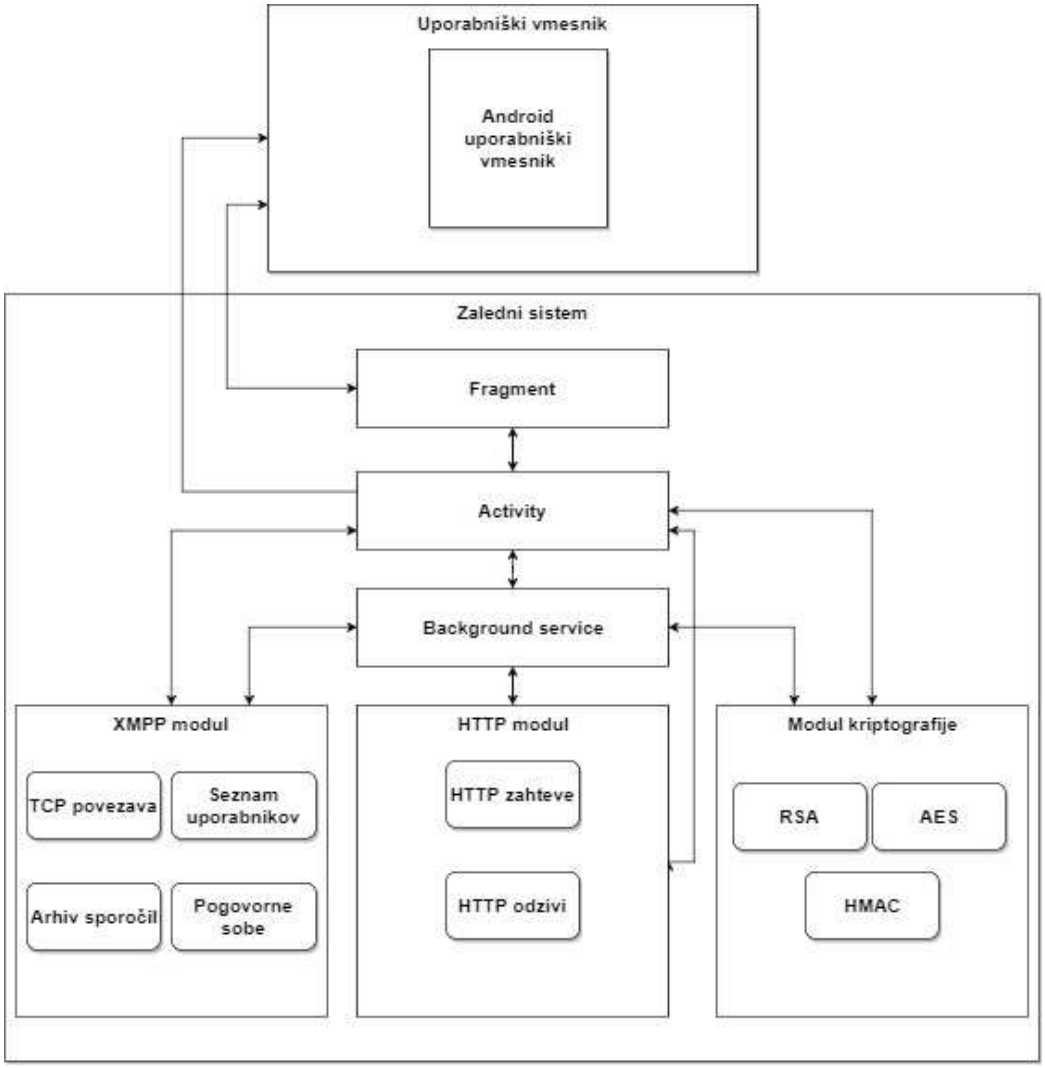
Naprave z naloženo aplikacijo za varno komunikacijo, bodo med seboj

komunicirale posredno. To pomeni, da bo poslano sporočilo najprej prišlo do XMPP strežnika. Strežnik bo nato preveril, če so naprave prejemnikov dosegljive in šele nato poslal sporočilo do naprav. Če je naprava prejemnika nedosegljiva, bo XMPP strežnik sporočilo zadržal in jo poslal, ko bo prejemnik zopet dosegljiv.

4.2 Arhitektura aplikacije

Android aplikacija je sestavljena iz uporabniškega vmesnika, ki je napisan v XML in zalednega sistema, ki je napisan v Javi.

Androidov uporabniški vmesnik sestavljajo View in ViewGroup objekti. ViewGroup objekti lahko vsebujejo več ViewGroup ali View objektov. Podrazredi teh objektov skrbijo za interakcijo in prikaz podatkov v različnih oblikah. Prikazani podatki so posredovani od Fragmentov in Aktivnosti. Aktivnost je osnovna komponenta Android aplikacije ter ključnega pomena za delovanje aplikacije. Obnašanje uporabniškega vmesnika v Aktivnostih predstavlja Fragmente, ki so uporabljeni za obnašanje različnih oken v uporabniškem vmesniku. Aktivnost skrbi za življenjski krog Fragmenta in prenos podatkov. Fragment je sposoben pošiljati in prejemati samo od Aktivnosti in od uporabniškega vmesnika. Ko je aplikacija odprta, aktivnost sprejema in pošilja podatke do vseh modulov, ki gradijo aplikacijo. Sprejemanje in oddajanje podatkov, ko je aplikacija zaprta, omogoča storitev, ki deluje v ozadju operacijskega sistema. Povezava med storitvijo in XMPP modulom skrbi, da se podatki prejemajo in pošiljajo, ko je aplikacija zaprta. Ko je to potrebno, storitev uporablja tudi HTTP modul in modul kriptografije. Arhitektura aplikacije je prikazana na sliki 4.1.



Slika 4.2: Arhitektura aplikacije.

4.3 Načrt aplikacije

4.3.1 Integracija podatkov

Pogovorne skupina, pogovor in uporabnik so objekti, ki predstavljajo sredstva iz zunanje podatkovne baze, ki so tesno povezani z XMPP instancami za uporabnike in pogovorne sobe [18]. Instanca pogovorne sobe predstavlja sku-

pinski pogovor, kjer lahko skupina uporabnikov komunicira. Uporabljena so imena, ki povezujejo XMPP instance z njihovimi sklopi podatkov v zunanji podatkovni bazi. Pogovorne skupine in pogovorne sobe povezuje identifikacijska številka pogovorne skupine, ki je tudi ime pogovorne sobe. Pogovorna soba loči pogovore po številki niti, ki je enaka identifikacijski številki pogovora na zunanji podatkovni bazi. Pri uporabnikih uporabniško ime v zunanji podatkovni bazi predstavlja poštni naslov, kar predstavlja problem pri XMPP uporabniškem imenu. Afna se tako pri poštnem naslovu kot tudi pri XMPP uporabniškem imenu uporablja za ločitev imena od domenskega naslova, zato je pri XMPP uporabniškem imenu afna, ki se nahaja v poštnem naslovu, zamenjana s kodo, ki predstavlja afno v URL nabornem jeziku.

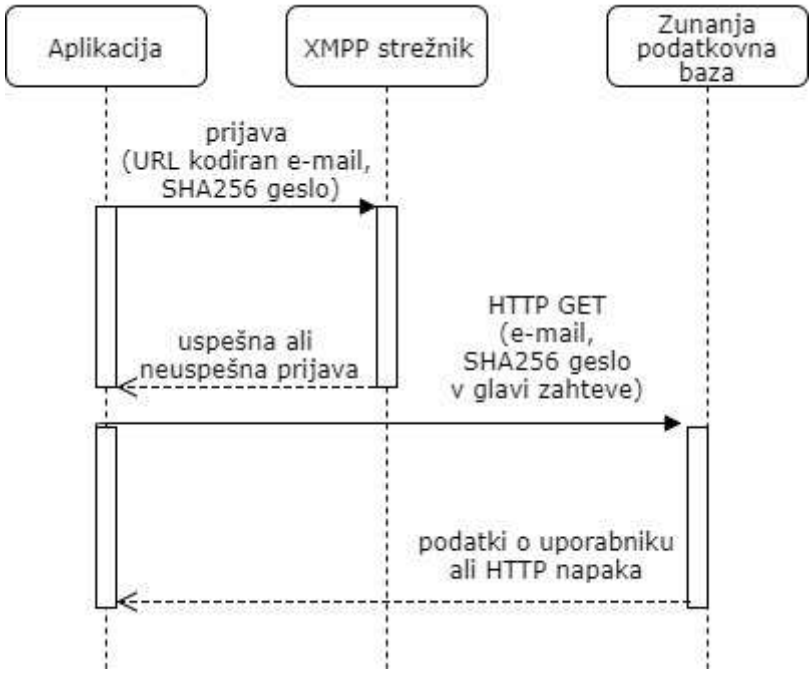
4.3.2 Preverjanje pristnosti

Pri varni komunikaciji je potrebno najprej preveriti pristnost uporabnika aplikacije. V našem prototipu aplikacije je to potrebno preveriti pri XMPP strežniku in zunanji podatkovni bazi.

Aplikacija najprej preveri pristnost pri zunanji podatkovni bazi, to stori s HTTP GET zahtevo, ki ima v glavi dodano lastnost za avtorizacijo z uporabniškim imenom in geslom, ki je v SHA256 razpršeni obliki. Zahteva naredi poizvedbo po uporabniku. Če je uporabnik vrnjen uspešno, je preverjanje opravljeno. Po preverjanju pri zunanji podatkovni bazi se zgodi prijava v XMPP strežnik, kjer je v uporabniškem imenu afna zamenjana z URL kodo, geslo pa je v SHA256 razpršeni obliki. Šele po uspešnem preverjanju pri XMPP strežniku lahko uporabnik začne uporabljati aplikacijo.

4.3.3 Delovanje objektov

Aplikacija podpira upravljanje z uporabnikom, in sicer lahko uporabnika ustvari. Pri kreiranju uporabnika se generirata privatni in javni ključ RSA za uporabnika, ki sta shranjena skupaj z ostalimi informacijami o uporabniku. Obstoječega uporabnika se lahko odstrani in tako prijava za odstranjenega



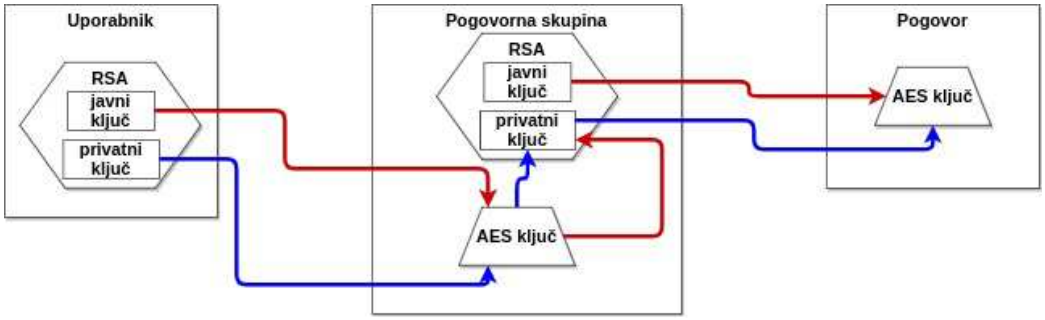
Slika 4.3: Potek preverjanja pristnosti.

uporabnika ni več mogoča.

Komuniciranje med uporabniki omogočajo pogovorne skupine, ki jih uporabnik lahko ustvari in vanjo doda ali odstrani druge uporabnike. Pri ustvarjanju skupine se generirata dve vrsti ključev, simetrični AES ključ in asimetrični RSA par privatnega in javnega ključa. AES ključ je shranjen v šifrirani obliki in ga uporabnik lahko dešifrira s svojim privatnim ključem, dešifriran AES ključ pa je uporabljen za dekriptiranje privatnega ključa pogovorne skupine. Pogovorno skupino lahko uporabnik tudi odstrani. Za dodajanje in odstranjevanje uporabnikov ter odstranitev skupine so potrebne pravice lastnika skupine. Uporabnik lahko odstrani tudi samega sebe, vendar ne sme biti zadnji uporabnik s pravicami lastnika skupine.

Preden uporabnik lahko pošlje sporočilo drugemu uporabniku, mora ustvariti pogovor. Pogovorov je v skupini lahko več. Pogovor vsebuje tudi šifrirano obliko simetričnega ključa AES, ki ga lahko dešifrira samo uporabnik z dešifriranim privatnim ključem pogovorne skupine. Dešifriran AES ključ je uporabljen za

šifriranje podatkov, ki jih uporabniki pošiljajo med seboj.



Slika 4.4: Prikaz delovanja kriptografije. Rdeče črte predstavljajo enkripcijo, modre dekripcijo.

4.3.4 XMPP razširitev Muc-Sub

Ker zaradi zunanjih dejavnikov stanje povezave pri mobilnih napravah niha, so običajne XMPP pogovorne sobe neprimerne, ker uporabnik zapusti sobo vedno, ko je nedosegljiv. To pomeni, da sporočila, ki so bila poslana v sobi med tem časom, ne pridejo do uporabnika, zato je uporabna razširitev pogovorne sobe z naročanjem. Gre za mešanico XMPP razširitve pogovorne sobe in razširitve za objavljanje in naročanje oziroma "Pub-Sub", tako pride tudi do imena "Muc-Sub"[20]. Pri tej razširitvi uporabnikom ni potrebno vstopiti v pogovorno sobo, da bi prejeli sporočila, biti morajo zgolj naročeni na to skupino. V primeru, da so nedosegljivi, se sporočila shranijo na XMPP strežniku in se dostavijo, ko je uporabnik zopet dosegljiv. Za prejemanje teh sporočil potrebujemo le enega poslušalca za sporočila, ki se sproži, ko sporočilo naslovljeno na prijavljenega uporabnika prispe do aplikacije. Iz sporočila se nato razbere, za katero pogovor in pogovorno skupino gre. Tudi za pošiljanje sporočil ni potrebno vstopiti v skupino, pač pa se uporabi metodo za pošiljanje sporočil. V sporočilu mora biti definiran prejemnik, to je pogovorna skupina, kateri je sporočilo namenjeno. Za potrebe naročanja in

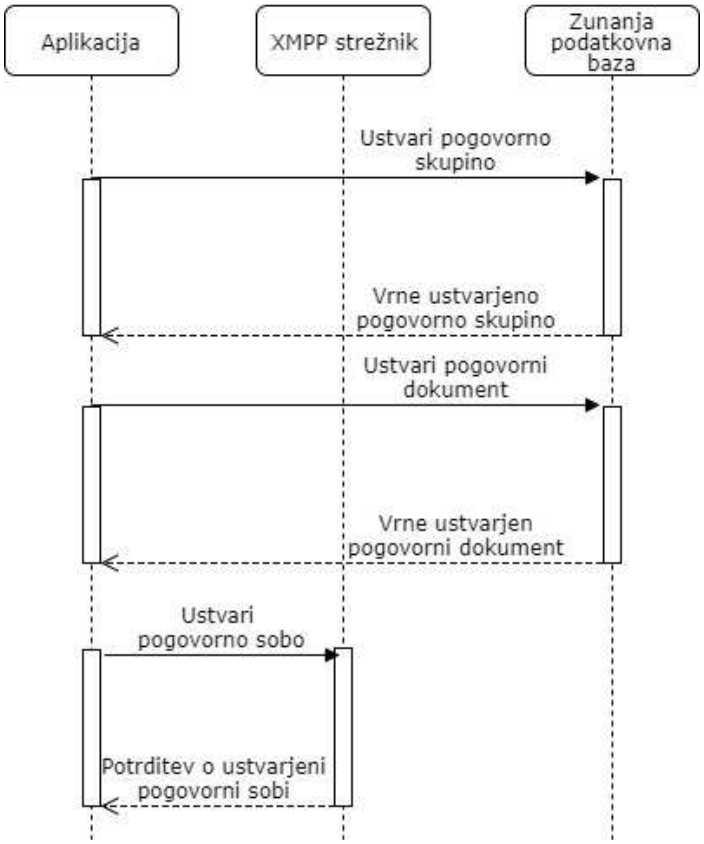
preklic naročnine je potrebna implementacija posebnih objektov, ki predstavljajo XMPP pakete za naročanje oziroma prekinitev naročnine.

4.3.5 Pošiljanje in prejemanje sporočil

Aplikacija podpira pogovorne skupine z naročanjem. Tudi ko poteka pogovor med samo dvema uporabnikoma, se ustvari pogovorna skupina, saj tako uporabimo razširitev in za sporočila, ki so namenjena nedosegljivim uporabnikom, ni potrebno dodatno skrbeti.

Za pošiljanje sporočil med uporabniki je potrebno ustvariti pogovorno skupino in pogovorni dokument na zunanji podatkovni bazi ter pogovorno sobo za več uporabnikov na XMPP strežniku. Člane pogovorne skupine se določi pri ustvarjanju pogovorne skupine. Ko je pogovorna skupina uspešno ustvarjena, lahko ustvarimo še pogovor, ki je sestavljen iz pogovornega dokumenta, ki je shranjen na zunanji podatkovni bazi, in pogovorne sobe za več uporabnikov na XMPP strežniku. Ime pogovorne sobe je sestavljeno iz identifikacijske številke pogovorne skupine in domene strežnika. Pogovorni dokument je določen z identifikacijsko številko dokumenta, ki se nahaja v niti vsakega poslanega sporočila.

Uporabnik pošlje sporočilo tako, da iz naloženega pogovornega dokumenta dešifrira AES ključ in s tem ključem šifrira telo sporočila. Pri vsakem sporočilu se generira inicializacijski vektor, ki skupaj s šifriranim telesom sporočila in AES ključem določijo HMAC vrednost. Inicializacijski vektor in HMAC vrednost sta dodana v sporočilo kot vsebina razširljivega elementa. Prejemnik sporočila potrebuje pogovorni dokument, do katerega lahko pride s pomočjo niti sporočila, ki vsebuje identifikacijsko številko dokumenta. Z uporabo ključa in inicializacijskega vektorja nato generira HMAC vrednost, ki se primerja z HMAC vrednostjo iz sporočila. Enaka HMAC vrednost potrdi integriteto sporočila in na vrsti je dešifriranje telesa sporočila, ki je dešifrirano z uporabo AES ključa.



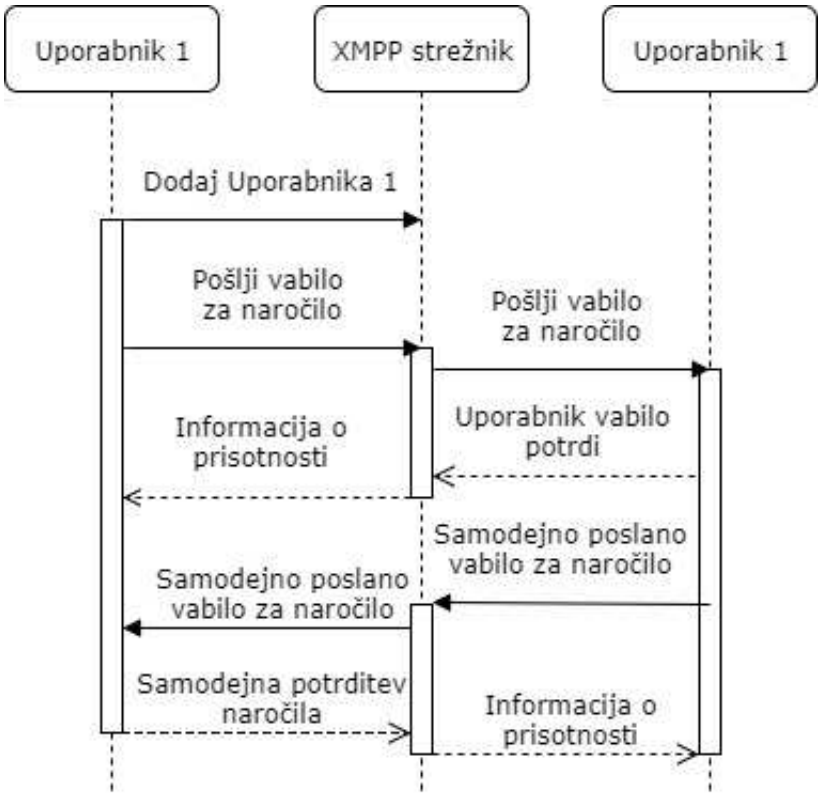
Slika 4.5: Sekvenčni diagram poteka ustvarjanja pogovora.

4.3.6 Seznam uporabnikov

Seznam uporabnikov prikazuje uporabnike in njihov status. Uporabnik lahko doda drugega uporabnika tako, da mu pošlje prošnjo. Povabljen uporabnik ima status "prošnja v čakanju", dokler povabila ne sprejme, nato je status lahko dosegljiv ali nedosegljiv. Uporabnik je lahko s seznama uporabnikov tudi odstranjen. Odstranjen uporabnik ne prejema in ne oddaja več informacije o prisotnosti.

Za implementacijo seznama je uporabljena XMPP razširitev "Roster"[17], ki deluje tako, da ima vsak uporabnik svoj seznam uporabnikov, na katere se lahko naroči. Da prilagodimo delovanje razširitve našim potrebam, je potrebno dodati nekaj samodejnih odzivov in korakov. Dodajanje uporabnika

na seznam in naročanje na informacije o prisotnosti uporabnika je v razširitvi ločeno, v našem primeru pa se to zgodi v istem sklopu. Uporabnik je najprej dodan na seznam, nato se mu pošlje še prošnjo. Prošnjo zavrne ali sprejme. Če jo sprejme, potem pošlje enako prošnjo nazaj in prvotni uporabnik nato samodejno sprejme prošnjo. Pri odstranjevanju uporabnika je potreben samo en korak, saj pri odstranjevanju uporabnika s seznama XMPP razširitev samodejno prekliče naročilo na informacije o prisotnosti uporabnika.



Slika 4.6: Sekvenčni diagram poteka dodajanja uporabnika.

Poglavje 5

Razvoj aplikacije

5.1 Integrirano razvojno okolje

5.1.1 Android Studio

Pri izdelavi prototipa je bil za razvijanje izbran Android Studio [12]. Gre za uradno integrirano razvojno okolje, ki je namenjeno izdelavi Android aplikacij. Zgrajeno je na podlagi integriranega razvojnega okolja IntelliJ IDEA in je posebej prilagojeno za razvijanje Android aplikacij. Android Studio je bil izbran, ker zajema vse potrebne tehnologije in lastnosti za razvoj prototipa. Podprta je Java, v katerem je napisana večina izvorne kode, in Gradle, ki je podprt za grajenje aplikacij. Poleg tega vsebuje še bogat urejevalnik za grafično uporabniške vmesnike na podlagi XML jezika. Prav tako je uporaben emulator za poganjanje in razhroščevanje Android aplikacij. Za grajenje aplikacij ima Android Studio podprto orodje Gradle.

Android Studio je na voljo na uradni spletni strani, kjer je na volji v različicah za Windows, Mac in Linux. Pri nameščanju se obenem prenese še Android programski razvijalni komplet ali SDK z najnovejšimi posodobitvami. Posodobitve, platforme in orodja lahko dodajamo ali odstranjujemo s pomočjo SDK Managerja. Za pisanje programske kode je potrebna Java, lahko se uporablja tudi novo podprti jezik Kotlin, vendar smo za našo di-

plomsko nalogo uporabili Javo. Java razvijalni komplet 8 ali JDK je potreben za uporabo Android Studio, saj je v tem razvijalnem okolju podprta Java 7 z nekaj določenimi stvarmi iz Java 8.

Ko imamo pripravljene in nameščene vse potrebne tehnologije in orodja, lahko ustvarimo nov projekt. Izberemo ime aplikacije in domeno paketa oziroma podjetja. Dodamo lahko še podporo za C++, vendar tega v našem primeru nismo potrebovali. Prav tako lahko spremenimo lokacijo, kamor se bo projekt shranil. V naslednjem koraku izberemo vrsto naprav, katerim bo aplikacija namenjena. Na izbiro so mobilne naprave ali tablice, pametne ure, televizijski sprejemniki in Android Auto. Za naše potrebe smo izbrali mobilne naprave ali tablice. Izbrati je potrebno tudi najnižjo verzijo Androidovega programskega razvijalskega kompleta, manjša kot je izbrana verzija, več naprav je podprtih, vendar je podprtih manj funkcij. V našem primeru smo izbrali verzijo 15 in tako podprli vse naprave, ki so aktivne na Google Play. Ko je izbrana še verzija, se projekt zgradi in s pisanjem izvorne kode lahko začnemo.

5.2 Razvoj zalednega sistem

5.2.1 Izbira programskega jezika

Pri izbiri programskega jezika praktično nismo imeli izbire, saj implementacijo vseh tehnologij in ogrodij omogoča le Java [14]. Glavno zahtevo je prestavljal Android Studio, ki za izdelavo aplikacij zahteva Javo. Ker je Java razširjen jezik, nismo imeli težav pri uporabi drugih tehnologij, saj obstaja že veliko knjižic, ki so na voljo za uporabo v Javi.

5.2.2 Namestitev XMPP klienta

Da lahko aplikacija komunicira z XMPP strežnikom, potrebuje XMPP klienta. Najbolj razširjen, uporabljen, razvit in primeren za uporabo je Smack klient [19]. Smack je odprtokodni klient, ki je napisan v Javi in je tako podprt

za Javo ter tudi za Android.

Smack klient se doda v projekt s Gradle konfiguracijo. V konfiguracijsko datoteko Gradle se doda odvisnosti Smack klienta, ki so potrebne. Pri odvisnosti se določi skupino, ime in verzijo knjižice. Za izdelavo aplikacije z varno komunikacijo so potrebne tri knjižnice Smack klienta. Smack Android-Extensions knjižnica vsebuje dodatne funkcionalnosti, ki jih omogoča Android. Smack TCP skrbi za TCP povezavo in Smack Experimental, ki vsebuje nove funkcionalnosti XMPP-ja, kot je arhiviranje sporočil. Te knjižnice so ključne pri izdelavi aplikacije za varno komunikacijo med prenosnimi napravami.

```
XMPPTCPConnectionConfiguration.Builder configBuilder =
    XMPPTCPConnectionConfiguration.builder();

configBuilder.setSecurityMode(ConnectionConfiguration.SecurityMode.required);
configBuilder.setCustomSSLContext(createSSLContext());
configBuilder.setXmppDomain(domain.resource);
configBuilder.setPort(PORT);

XMPPTCPConnection.setUseStreamManagementResumptionDefault(true);
XMPPTCPConnection.setUseStreamManagementDefault(true);

AbstractXMPPConnection connection = new
    XMPPTCPConnection(configBuilder.build());

connection.connect()
connection.login(username, password);
```

Kazalo programskih kod 5.1: Primer prijave na XMPP strežnik s Smack klientom

Implementacija Smack klienta v aplikacijo je preproste oblike. Določiti je potrebno naslov gostitelja ter vrata. Da je poskrbljeno za varno komu-

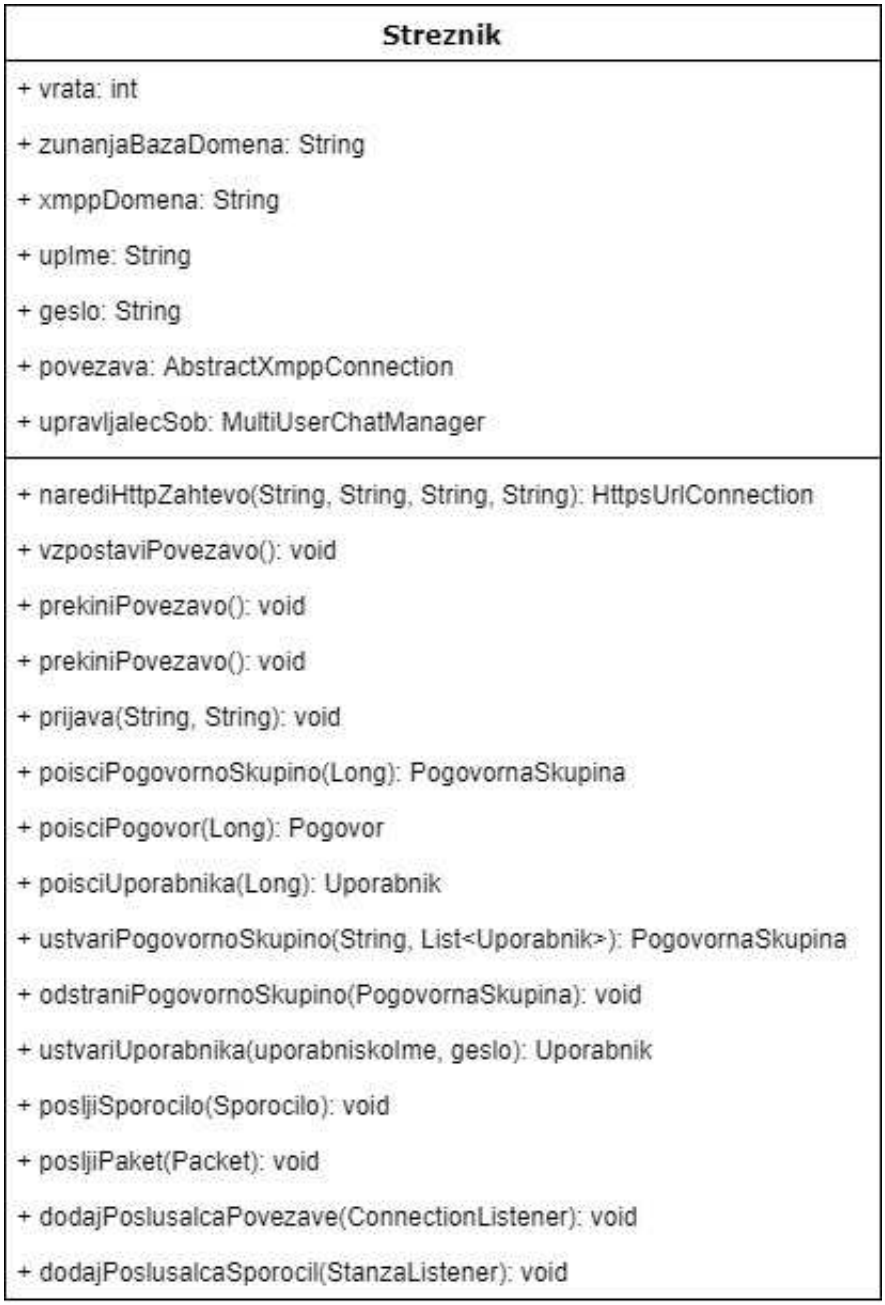
nikacijo, se omogoči varni način, pri tem je potrebno dodati še preverjanje certifikatov in upravljanje pretoka.

5.2.3 Implementacija objekta strežnika

Centralni objekt, ki povezuje vse objekte, se imenuje strežniški objekt. Objekt je sestavljen iz komponente, ki določa vrata za XMPP strežnik in še dveh domenskih komponentah, ki vsebujeta informacijo o naslovu zunanje podatkovne baze ter XMPP strežnika. Za potrebe avtentikacije sta v strežniški objekt dodana tudi uporabniško ime in geslo. Centralni objekt vsebuje še dve komponenti, ki pa sta del Smack klienta. Prva komponenta skrbi za povezavo z XMPP strežnikom, imenujemo jo komponenta povezave, druga komponenta pa je za upravljanje z XMPP pogovornimi sobami in jo imenujemo komponenta za upravljanje sob.

Pomembna metoda centralnega objekta, ki se uporablja tudi v objektih naše aplikacije, je metoda za izgradnjo HTTP zahtev. Metoda sprejme URL, metodo, tip poslane vsebine in tip prejete vsebine kot parametre ter na podlagi teh oblikuje HTTP zahtevo. Ker gre v našem primeru za varno aplikacijo, je uporabljen "HttpsURLConnection"objekt, ki vsebuje tudi protokol za varnost. Za dodatno varnost so preverjeni tudi certifikati, če so izdani od preverjenih ponudnikov. Če objekt vsebuje uporabniško ime in geslo, sta ti dve vrednosti dodani v glavo zahteve kot lastnost običajne avtorizacije. Geslo je podano v SHA256 razpršeni vrednosti. Metoda nato vrne zgrajen "HttpsURLConnection"objekt, ki se ga lahko izvrši.

Za upravljanje povezave z XMPP strežnikom ima centralni objekt metodi za vzpostavitev in prekinitev povezave. Da metode v tem objektu in tudi ostalih postanejo uporabne, je potrebna avtentikacijo, za katero poskrbi metoda za prijavo. Metoda je sestavljena iz dveh delov, najprej se zgodi preverjanje pristnosti z zunanjo bazo, kjer se izvede HTTP zahteva za uporabnika in nato še prijava v XMPP strežnik, za katero poskrbi Smack klient. Kot se preverja certifikate pri zunanji bazi, se preverjajo certifikati tudi pri XMPP strežniku. Uporabne so metode za iskanje pogovornih skupin, pogo-



Slika 5.1: Entiteta objekta Streznik

vorov in uporabnikov, ki za parameter sprejmejo identifikacijsko številko in nato vrnejo najden objekt, če obstaja. Vsa sporočila se pošiljajo z uporabo metode za pošiljanje sporočil, ki preko komponente za povezavo z XMPP

strežnikom pošlje XMPP sporočilo. Za pošiljanje paketov za naročanje na skupine je metoda enaka prejšnji, le da ta sprejme paket in ne sporočilo.

Za ustvarjanje pogovorne skupine in tudi pogovorne sobe je zadolžena metoda, ki uporabi metodo za shranjevanje pogovorne skupine, ki ustvari pogovorno skupino na zunanji podatkovni bazi in nato s pomočjo upravljalca pogovornih sob ustvari še pogovorno sobo z imenom, ki je sestavljeno iz identifikacijske številke pogovorne skupine. Obratno operacijo opravi metoda za odstranjevanje pogovorne skupine, ki najprej odstrani skupino na zunanji podatkovni bazi in nato še sobo na XMPP strežniku. Objekt vsebuje še metodo za ustvarjanje uporabnika, ki preko parametrov sprejme uporabniško ime in geslo ter vrne ustvarjenega uporabnika. Metoda z uporabo gesla generira privatni ključ in nato še javni ključ, oba se združita v uporabniški ključ.

Za nadzorovanje XMPP povezave je lahko s pomočjo metode dodan poslušalec za povezavo iz Smack klienta, ki se sproži ob vseh spremembah povezave. Smack klient prav tako nudi poslušalca za vse pakete, ki je v naši metodi prilagojen s filtrom, ki prepušča samo sporočila, ki ustrezajo merilom aplikacije. Merilo zahteva, da gre za sporočilo, ki mora vsebovatei dodatne vrednosti, kot sta inicializacijski vektor ter HMAC vrednost.

5.2.4 Implementacija objekta pogovorne skupine

Pogovorna skupina je objekt, ki vsebuje identifikacijsko številko tipa, ki je uporabljena za poizvedbe skupin in je tudi sestavni del imena XMPP pogovorne sobe [18]. Strežniška komponenta nudi dostop do strežniških metod. Skupini sta lahko dodana ime in opis za lažje razlikovanje med skupinami. Ključni elementi skupine so seznam uporabnikov, pogovorna soba za več uporabnikov ter asimetrični in simetrični ključi. Objekt ima paketno privatni konstruktor, ki zahteva strežniško komponento, ime in seznam uporabnikov.

Poleg metod za pridobivanje in nastavljanje vrednosti komponent so potrebne dodatne metode, ki so ključne za delovanje pogovornih skupin. Metodi za dodajanje in odstranjevanje poskrbita za upravljanje z uporabniki

Slika 5.2: Entiteta objekta PogovornaSkupina

v pogovorni sobi in na zunanji podatkovni bazi. Pri metodi za dodajanje s parametri določimo uporabnika in njegovo vlogo v skupini, lahko je administrator ali navaden uporabnik. Metoda preveri, če ima trenutni uporabnik vlogo administratorja in šele nato izvede dodajanje. Pri odstranjevalni metodi z edinim parametrom določimo uporabnika za odstranitev, tudi pri tej metodi je preverjeno, če operacijo izvaja administrator. Da lahko uporabnik zapusti pogovorno skupina, je na voljo še metoda za zapustitev skupine. Za spreminjanje vloge uporabnikov obstajata dve metodi, tj. za dodelitev in

odvzem administratorskih pravic. Obe metodi preverita, če operacije izvaja administrator. Da se lahko komunikacija začne, je potrebna še možnost za ustvarjanje pogovora, ki ustvari in vrne ustvarjen pogovorni objekt. Za pridobitev že ustvarjenih pogovorov je potrebna metoda, ki vrne seznam pogovor v skupini.

Pogovorna skupina je objekt, katere podatki so shranjeni v zunanji podatkovni bazi, kot je JSON dokument, zato objekt vsebuje metodo, ki zna prebrati te podatke in metodo, ki zna zapisati te podatke v dokument. Ti dve metodi sta privatni, a sta uporabljeni v metodah za shranjevanje, posodabljanje in brisanje pogovorne skupine, ki so javne. Metoda za shranjevanje najprej generira vse ključe in jih skupaj z ostalimi podatki pretvori v JSON vsebino ter to vsebino nato kot objekt s pomočjo HTTP POST zahteve pošlje na strežnik in nato odziv strežnika, ki je v JSON obliki, prebere. Metodi za posodabljanje sta dve, ena posodobi podatke na strežniku, druga pa v aplikaciji. Metoda, ki posodablja podatke na strežniku, je enaka metodi za shranjevanje, vendar uporablja HTTP PUT zahtevo. Pri posodabljanju podatkov v aplikaciji, gre za HTTP GET zahtevo, ki pridobi podatke iz strežnika in jih shrani v objekt. Pri metodi za brisanje gre za HTTP REMOVE zahtevo.

5.2.5 Implementacija objekta pogovora

Implementacija objekta Pogovor je ključnega pomena, saj vsebuje simetrični ključ, ki se uporablja za šifriranje sporočil, ki se pošiljajo med uporabniki v istem pogovoru. Poleg ključa pogovor vsebuje še identifikacijsko številko in ime. Konstruktor tega objekta je paketno privatni in zahteva strežniško komponento, pogovorno skupine in ime.

Objekt vsebuje metodo za šifriranje, ki preko parametra prejme objekt sporočila, ki vsebuje vsebino sporočila in vse ostale informacije. Metoda ta objekt šifrira in ga vrne kot objekt šifriranega sporočila, ki je skoraj enak prejšnjemu objektu, le da ima šifrirano vsebino in dodan generiran inicializacijski vektor in HMAC vrednost. Obratno operacijo naredi metoda za dešifriranje, ki preko parametra sprejme objekt šifriranega sporočila, prvo

Slika 5.3: Entiteta objekta Pogovor

preveri HMAC vrednost, nato dešifrira vsebino in vrne objekt običajnega sporočila. Poleg šifriranja objekta vsebuje še metodo za nalaganje datotek, ki preko parametra sprejme seznam datotečnih objektov in shrani datoteke, ki so v datotečnih objektih ter vrne isti seznam datotečnih objektov, čeprav objekti sedaj vsebujejo identifikacijske številke. Pomembna metoda objekta je tudi metoda za pošiljanje, ki prejme objekt sporočila kot parameter, sporočilu nato doda informacijo o pošiljatelju in prejemniku, ga šifrira s pomočjo metode za šifriranje sporočila in pošlje naslovniku preko strežniškega objekta. Za dostop do arhiva sporočil pogovora je zadolžena metoda za pridobitev seznama arhiviranih sporočil, ki kot parameter sprejme objekt poizvedbe za arhivirana sporočila in nato vrne seznam pridobljenih sporočil iz arhiva. Za pridobitev sporočil poskrbi Smack klient, ki vpraša XMPP strežnik po sporočilih, ki ustrezajo poizvedbi. Strežnik nato vrne sporočila, ki so v šifrirani obliki. Sporočila so vrnjena kot objekti Smack klienta, ki jih

nato metoda pretvori v objekt šifriranega sporočila in šele nato s pomočjo metode za dešifriranje vsa sporočila dešifrira in vrne seznam sporočil.

Objekt pogovora je po delovanju z zunanjo podatkovno bazo enak pogovorni skupini in vsebuje enake metode za shranjevanje, posodabljanje in brisanje. Razlike je samo pri metodi za shranjevanje, ki je v tem primeru paketno privatna, saj se uporablja v metodi za ustvarjanje pogovora v objektu pogovorne skupine.

5.2.6 Implementacija objekta uporabnika

Objekt uporabnika vsebuje identifikacijsko številko ter osnovne podatke o uporabniku, ime, naslov, spol in poštni naslov, ki je hkrati tudi uporabniško ime. Poleg tega pa vsebuje še javni in privatni ključ. Objekt uporabnika v konstruktorju zahteva strežniško komponento, poštni naslov in ime.

Slika 5.4: Entiteta objekta Uporabnik

Ker je uporabnik objekt, ki ima podatke shranjene v zunanji podatkovni bazi, vsebuje enake metode kakor pogovorna skupina in pogovor. Poleg teh

metod objekt uporabnika vsebuje še metodo za pridobivanje informacij o vlogi v pogovorni skupini, ki je podana v metodo kot parameter. Za uporabnost je dodana metoda, ki zgradi XMPP objekt imenovan Jid, ki je posebno oblikovano uporabniško ime za potrebe XMPP storitev.

5.2.7 Implementacija objektov sporočil

Sporočila so lahko šifrirana ali nešifrirana, zato sta implementirana dva objekta. Objekt za običajna sporočila je razširjen objekt šifriranega sporočila, kar pomeni da podeduje nekaj komponent, ki jih ima objekt za šifrirana sporočila. Oba objekta vsebujeta komponente za identifikacijsko številko običajnega in arhiviranega sporočila, številko niti, pošiljatelja in prejemnika, vsebino sporočila, datum in seznam identifikacijskih številk datotek, ki so pripete sporočilu. Šifrirano sporočilo vsebuje še inicializacijski vektor in HMAC vrednost.

Objekt šifriranega sporočila vsebuje dve metodi. Prva služi branju podatkov iz XMPP sporočila, druga pa kreiranju XMPP sporočila. Metoda za branje sprejme XMPP sporočilo preko parametra in prebere ter shrani podatke v objekt šifriranega sporočila. Druge metoda iz podatkov, ki so v objektu, zgradi XMPP sporočilo in ga vrne. Objekt običajnega sporočila ima metodo za pridobitev datotek, ki so pripete zraven sporočila. Ta metoda s pomočjo seznama identifikacijskih številk datotek pridobi dejanske datoteke z zunanje podatkovne baze.

Objekt šifriranega sporočila vsebuje konstruktor, ki zahteva XMPP sporočilo, ki ga s pomočjo metode za branje XMPP sporočila prebere in shrani podatke v objekt. Objekt običajnega sporočila ima dva konstruktorja, enega brez parametrov in enega s parametrom za vsebino.

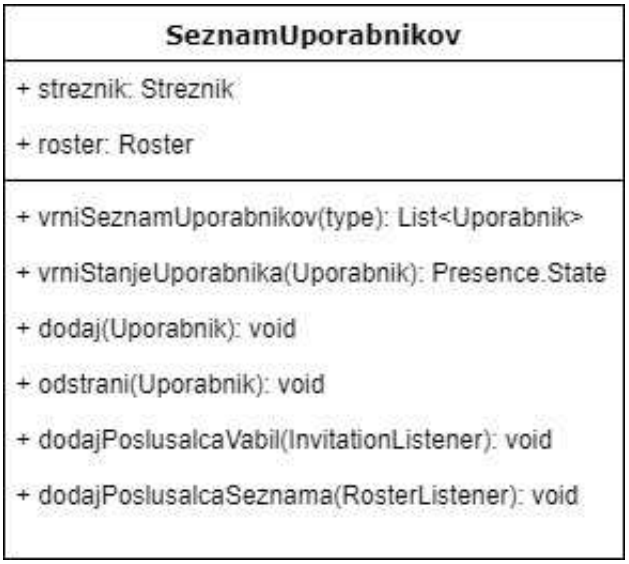
```
<message to="receiver@domain.si/"
        from="7301@conference.domain.si">
   <event xmlns="http://jabber.org/protocol/pubsub#event">
      <items node="urn:xmpp:mucsub:nodes:messages">
         <item id="18277869892147515942">
            <message lang="en" to="receiver@domain.si"
                from="7301@conference.domain.si/sender"
                type="groupchat" id="Sd31k-19">
               <archived xmlns="urn:xmpp:mam:tmp"
                   by="7301@domain.si" id="1504273045766922" />
               <stanza-id xmlns="urn:xmpp:sid:0" by="7301@domain.si"
                   id="1504273045766922" />
               <sec xmlns="sec:hmac_iv"
                   iv="e8c793ec5b9b2905971446ebf5b6acf5"
                   hmac="335350c...efd4cc6c55ca9713c1">
                  <attachment>13197</attachment>
               </sec>
               <body>3B4PPNU/GhGAUis+k4Xvzw==</body>
               <thread>12497</thread>
            </message>
         </item>
      </items>
   </event>
</message>
```

Kazalo programskih kod 5.2: Primer Muc-Sub sporočila [20]

5.2.8 Implementacija objekta seznama uporabnikov

Četudi se imenuje seznam uporabnikov, gre za več kot le seznam. Gre za objekt, ki vsebuje potrebne funkcionalnosti za upravljanje z uporabniki na seznamu in tiste uporabnike, ki še niso na seznamu. Objekt vsebuje

strežniško komponento in "Roster"komponento, ki je del Smack klienta. S konstruktorjem, ki zahteva strežniško komponento, se inicalizira tudi "Roster"komponenta, ki uporabi komponento za povezavo z XMPP strežnikom. V konstruktorju se doda tudi poslušalec za vabila, ki avtomatsko sprejema prošnje dodanih uporabnikov.



Slika 5.5: Entiteta objekta SeznamUporabnikov

Za pridobitev seznama uporabnikov se uporablja metoda, ki s pomočjo "Roster"[17] komponente pridobi seznam XMPP uporabnikov. Nato za vsakega uporabnika pridobi še informacije o uporabniku z zunanje podatkovne baze in ustvari objekt uporabnika ter ustvarjene objekte vrne v seznam. Naslednja je metoda, ki vrne stanje uporabnika, ki je podan v metodo preko parametra. Uporabnik je lahko dodan ali odstranjen s seznama uporabnikov z metodama za dodajanje ali odstranjevanje. Ti dve metodi sprejmeta objekt uporabnika preko parametra in ga pretvorita v XMPP vnos za seznam ter ga nato s pomočjo "Roster"komponente dodata ali odstranita. Pri dodajanju se pošlje prošnjo drugemu uporabniku, ki jo lahko sprejme z metodo za potrditev prošnje. Metoda najprej pošlje paket za potrditev prošnje in nato pošlje prošnjo uporabniku nazaj s pomočjo "Roster"komponente. Za

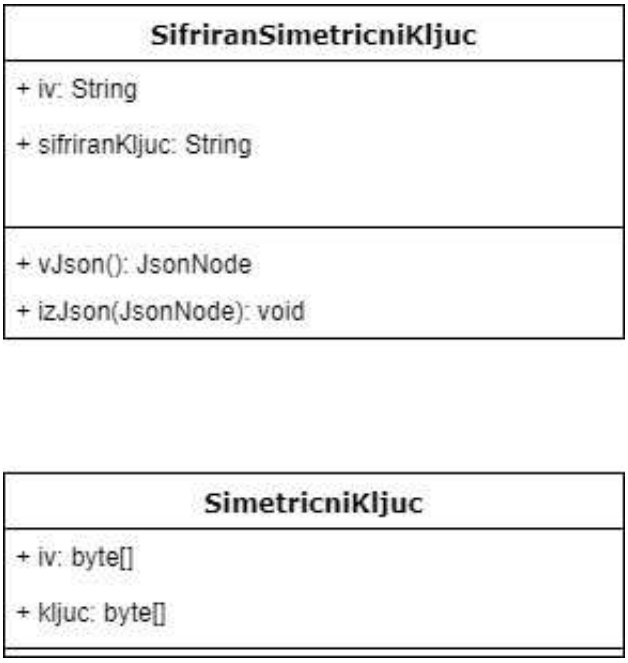
zavrnitev prošnje je metoda bolj enostavna, saj pošlje XMPP strežniku zgolj paket z informacijo o zavrnitvi prošnje. Poslušalec za prošnje je oblikovan tako, da se aktivira, ko prispe prošnja od uporabnika, ki ni še na seznamu uporabnikov in ne obstaja naročnina na tega uporabnika. Metoda za dodajanje poslušalca seznama uporabnikov omogoča dodajanje poslušalca, ki sliši spremembe, ki se zgodijo na seznamu.

5.2.9 Implementacija objektov kriptografije

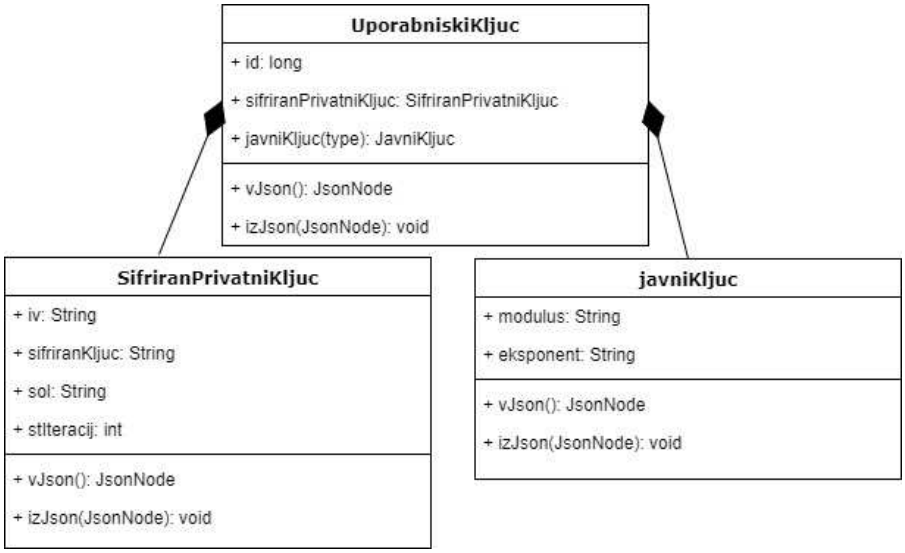
Za delovanje simetrične kriptografije [2] je potreben simetrični ključ, zato je potrebno implementirati objekt, ki predstavlja ta ključ. Objekt simetričnega ključa sestavljata vrednost ključa in inicializacijski vektor. Vse te vrednosti se poda že v konstruktorju tega objekta. V naši aplikaciji pride do potrebe tudi po šifriranem simetričnem ključu, zato je implementiran tudi objekt za te potrebe. Ta vsebuje šifrirano vrednost ključa in inicalizacijski vektor. Objekt šifriranega simetričnega ključa vsebuje tudi metode za pisanje in branje podatkov v JSON obliki, da se lahko ta ključ shrani v zunanjo podatkovno bazo.

Pri asimetrični kriptografiji [2] sta dva ključa, javen in privatni. V našem primeru se javen in privatni ključ nahajata v istem objektu, vendar je privatni ključ šifriran s simetričnim ključem, ki ga vsebujejo pogovorne skupine ali pa izpeljan iz uporabniškega gesla. Objekt uporabniškega ključa vsebuje objekta šifriranega privatnega in javnega ključa ter identifikacijsko številko. Tako kot šifriran simetrični ključ, tudi uporabniški ključ vsebuje metode za branje in pisanje podatkov v JSON obliki, saj se uporabniški ključ hrani v zunanji podatkovni bazi. Objekt šifriranega privatnega ključa je sestavljen iz inicializacijskega vektorja, šifrirane vrednosti ključa, soli in števila iteracij. Javni ključ je sestavljen iz komponent modulusa in eksponenta.

Objekt, ki je namenjen kriptografskim operacijam, je imenovan kriptografski modul. Modul vsebuje operacije za generiranje simetričnih in asimetričnih ključev, kot tudi operacije, ki se izvajajo na teh ključih. Pri generiranju simetričnega ključa se generirajo naključne vrednosti gesla in



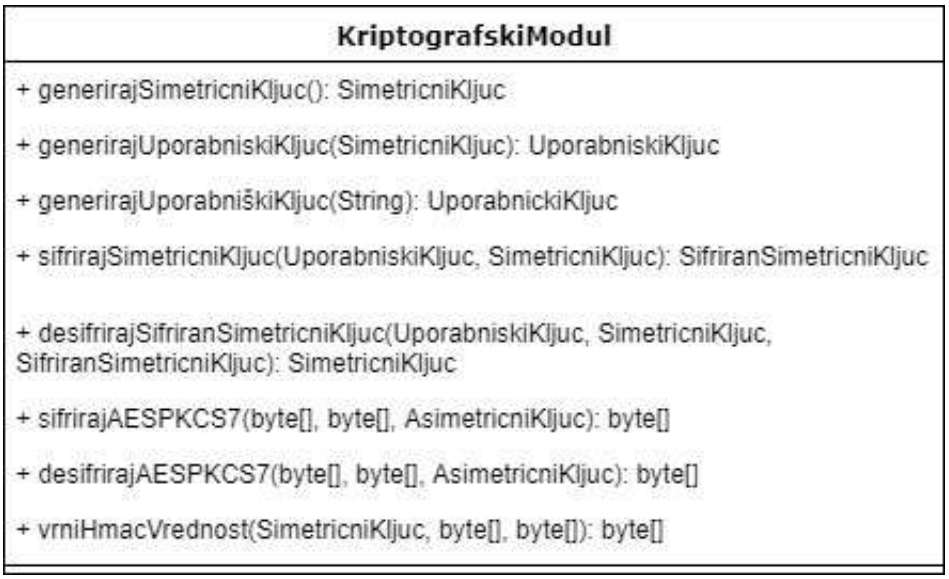
Slika 5.6: Entiteti objektov SimetricniKljuc in SifriranSimetricniKljuc



Slika 5.7: Entiteta objekta UporabniskiKljuc

soli, ki se nato s pomočjo funkcije za izpeljavo ključa iz gesla imenovana "pbkdf2"uporabita za izpeljavo vrednosti ključa. V konstruktor za simetrični

ključ se poda vrednost ključa skupaj z naključno generiranim inicializacijkim vektorjem in ključ je ustvarjen. Za generiranje uporabniškega ključa obstajata dve metodi, ki preko parametra sprejmeta geslo ali simetrični ključ. Metoda z geslom se uporablja pri uporabi uporabniškega ključa od uporabnika, metoda s simetričnim ključem pa pri uporabniškem ključu pogovorne skupine. Obe metodi delujeta podobno, le da se pri metodi z geslom najprej izpelje simetrični ključ iz gesla in nato sledi še ostalo. Za generiranje para privatnega in javnega dela vrednosti se uporabi knjižnico SpongyCastle [15]. Knjižnica generira naključni vrednosti modulusa in exponenta, ki sta podana v konstruktor in javni ključ je zgrajen. Potreben je še privatni del, ki ga je potrebno šifrirati. Vrednosti privatnega dela javni eksponent, modulus, privatni eksponent, P, Q, DP, DQ in Qinv se zapakirajo skupaj in pretvorijo v bajtni niz. Niz se nato šifrira in skupaj z naključno generiranim inicializijskim vektorjem in soljo poda v konstruktor za šifriran privatni ključ. Nato se uporabi konstruktor za uporabniški ključ, ki sprejme šifriran privatni in javni ključ. Uporabniški ključ je zgrajen.



Slika 5.8: Entiteta objekta KriptografskiModul

Kriptografski modul premore metode za simetrične in asimetrične opera-

cije kriptografije. Metoda za šifriranje z AES algoritmom sprejme simetrični ključ, inicializacijski vektor in vrednost, ki jo metoda zašifrira in vrne. Za obratno operacijo metoda sprejme iste parametre, le da metoda tokrat vrednost dešifrira in nato vrne. Pri obeh metodah se poleg AES algoritma za šifriranje uporablja tudi PKCS7 zapolnjevanje. Za pomočjo asimetrične kriptografije pa metode šifrirajo in dešifrirajo asimetrični ključ. Metoda za šifriranje simetričnega ključa sprejme preko parametra uporabniški ključ in simetrični ključ, ki se bo šifriral. Nato z javnim delom uporabniškega ključa šifrira simetrični ključ in ga vrne. Metoda za dešifriranje ključa sprejme šifriran simetrični ključ in uporabniški ključ ter geslo ali simetrični ključ. Metoda, ki sprejme geslo, iz njega izpelje simetrični ključ, s katerim se najprej dešifrira privatni ključ v uporabniškem ključu. Z dešifriranim privatnim ključem nato dešifrira podan šifriran simetrični ključ. Poleg metod za šifriranje vsebuje kriptografski modul tudi metodo za generiranje HMAC vrednosti. Metoda preko parametra sprejme objekt asimetričnega ključa, ki ga uporabi za generiranje HMAC vrednosti, ki jo metoda vrne. Za generiranje HMAC vrednosti je uporabljena knjižnica SpongyCastle [15].

5.2.10 Implementacija Android aplikacije

Uporabniški vmesnik je zgrajen iz glavne aktivnosti, treh fragmentov in servisa, ki deluje v ozadju. Aktivnost povezuje vse tri fragmente in servis ter vsebuje glavne funkcionalnosti aplikacije za povezovanje z XMPP strežnikom in zunanjo podatkovno bazo. Potrebne podatke nato posreduje fragmentom oziroma fragmenti pridobijo informacije preko aktivnosti. Ko servis generira obvestila, v obvestilo doda informacijo v obliki Intenta, naj se zažene glavna aktivnost in aktivnost nato prikaže ustrezen fragment. Aplikacija ob izklopu pošlje informacijo servisu, da lahko prične s poslušanjem. Ker se aplikacija lahko nenadoma zapre, servis v intervalih preverja, če je aplikacija odprta.

5.3 Delovanje uporabniškega vmesnika

Prvo okno aplikacije, ki se prikaže uporabniku, je okno za prijavo. Ob začetku aplikacije se inicializira strežniški objekt, nato pa se izvede metoda za začetek povezave z XMPP strežnikom. Dodan je tudi poslušalec za sporočila, ki ob prejetju sporočila pokaže obvestilo v vrstici za obvestila. Ob kliku na to obvestilo se odpre okno z ustreznim pogovorom. Poleg strežniškega objekta se inicalizira tudi objekt seznama uporabnikov, dodana sta tudi poslušalca za avtomatsko potrjevanje prošenj in poslušalec za prošnje uporabnikov, ki niso še dodani na seznam. Poslušalec za prošnje uporabnikov ob prejetju prošnje prikaže na zaslonu opozorilni dialog, kjer je izpisan uporabnik, ki pošilja prošnjo; na voljo sta opciji za potrditev in zavrnitev prošnje. Vsi ti poslušalci so dodani preden uporabnik opravi preverbo prisotnosti, ker obstaja možnost, da bodo podatki prišli takoj po preverbi. Uporabnik je vprašan po uporabniškem imenu in geslu. Uporabnik vnese podatke in s pritiskom na gumb za prijavo sproži metodo za preverjanje pristnosti v strežniškem objektu. Uporabnik lahko izbere tudi opcijo za registracijo novega uporabnika, kjer vnese svoje ime, poštni naslov in geslo.

Po uspešni prijavi se prikaze okno s seznamom uporabnikov. Preden se pojavi seznam uporabnikov, je dodan še poslušalec za spremembe v seznamu uporabnikov, ki ob prejetju obvestila spreminja podatke prikazane na oknu. Poleg dodajanja in odstranjevanja uporabnikov na seznamu se osvežuje tudi status posameznega uporabnika. Ko je poslušalec dodan, se kliče še metoda za pridobitev dejanskega seznama uporabnikov, rezultat pa se prikaže v oknu. Uporabnik lahko v tem oknu izbira med različnimi opcijami, kot so dodajanje in odstranjevanje uporabnikov, kjer se uporabljajo metode iz objekta seznama uporabnikov. Za ustvarjanje pogovornih skupin z več uporabniki uporabnik izbere opcijo za ustvarjanje skupin in v opozorilnem dialogu izbere, katere uporabnike hoče dodati v skupino. Te skupine so prikazane na koncu seznama pod uporabniki. Uporabnik lahko tudi zbriše svoj račun, tu se uporabi metoda za odstranjevanje v objektu uporabnika ali pa se lahko odjavi, v tem primeru se kliče metoda za prekinitev povezave v strežniškem

objektu. Če pritisne na enega izmed uporabnikov, se odpre okno za pogovor.

Preden se prikaže okno za pogovor, se naprej pregleda, če je že ustvarjena skupina za pogovor. Nato se poišče pogovor z metodo za pridobivanje seznama pogovorov. Če pa skupina ali pogovor ni najden, se najprej ustvari objekt skupina in nato še objekt pogovora. Ko je objekt pogovora pripravljen, se okno prikaže. Doda se tudi nov poslušalec za sporočila, ki prejeta sporočila tokrat prikazuje v trenutnem oknu, prejšnjega poslušalca pa se odstrani. V oknu za pogovor lahko vidi zadnjih nekaj sporočil, ki so bila naslovljena v ta pogovor. Uporabnik ima opcijo, da zahteva zgodovino sporočil iz tega pogovora. Uporabi metodo za pridobitev sporočil iz arhiva v objektu pogovora. Če hoče poslati sporočilo drugim uporabnikom tega pogovora, napiše sporočilo in ga pošlje s pritiskom na gumb. Pri tem se zgradi nov objekt sporočila, ki se nato še šifrira in nato pošlje s klicem metode za pošiljanje v objektu pogovora. Poleg besedila lahko pošlje tudi datoteke z naprave. Najprej izbere opcijo za pošiljanje datotek in nato izbere želene datoteke. Te datoteke se z metodo za nalaganje šifrirajo in naložijo na zunanjo podatkovno bazo, identifikacijske številke se dodajo kot razširitev v objekt sporočila, sporočilo pa se pošlje uporabniku. Aplikacija uporabnika prejme identifikacijske številke datotek, ki jih naloži z zunanje podatkovne baze, dešifrira in prikaže v oknu za pogovor. Uporabnik lahko doda in odstrani uporabnike v pogovoru, če si lasti pravice administratorja skupine. Uporabnikom lahko tudi doda ali odvzame pravice administratorja. Pri tem se uporabijo metode iz objekta pogovorne skupine za dodajanje ali odstranjevanje uporabnikov ter metodi za dodajanje ali odvzem administratorskih pravic. Poleg dodajanja uporabnikov je lahko dodan tudi nov pogovor. Pogovorno skupino lahko uporabnik tudi zapusti, v tem primeru se kliče metoda za zapustitev pogovorne skupine.

V ozadju aplikacije ves čas deluje servis, ki se zažene že ob priklopu naprave. Servis v času delovanja dejanske aplikacije miruje, ko pa se aplikacija zapre, servis začne s pomočjo poslušalcev čakati na sporočila. Ko XMPP sporočilo prispe, servis naloži ustrezen objekt pogovora in pogovorne sku-

pine ter s pomočjo ključev sporočilo dešifrira in ga prikaže kot obvestilo v vrstici za obvestila.

5.4 Razvoj testiranja modulov

Testiranje modulov je pri pisanju programske kode za aplikacije, ki vsebujejo več objektov, zelo pomembno. Testi so nujni za vzdrževanje programske kode, saj rezultati testov nudijo informacijo o tem, kateri moduli delujejo in kateri ne. Po vsakem sklopu sprememb kode je priporočljivo pognati teste in preveriti, da se s spremembo ni kaj pokvarilo [1].

Testi vsebujejo svoj razred za vsako implementacijo našteto zgoraj in v tem razredu je napisan test za vsako metodo. Metode testov, ki testirajo metode implementacij, so imenovane tako, da se že po imenu razume, kaj določena metoda testira. Število metod v testnem razredu je enako ali večjo kot število metod v razredu implementacije, ker se metoda lahko obnaša na več načinov, in sicer glede na vnosne parametre. Testne metode preverjajo točnost podatkov s trdilnimi metodami, kjer je s parametrom vnaprej določen rezultat in podatek, ki ga določena implementacijska metoda vrne. Trdilna metoda ob nepričakovanih podatkih testno metodo ustavi in javi napako z opisom, kakšna je bila pričakovana informacija in kakšna je bila podana. Eden izmed načinov obnašanja metode je odziv na napake. Metode v testnem razredu v tem primeru pričakujejo vrženo izjemo od metod v implementacijskem razredu. Pri testiranju modulov je priporočljivo, da so testi med seboj neodvisni, zato se lahko vsi testni razredi in metode poganjano neodvisno, kar dodatno pripomore k točnosti rezultatov.

5.5 Uporaba sistema Git

Pri pisanju izvorne kode je uporaba Git-a [23] dandanes že nepogrešljiva. Za izdelavo opisane aplikacije so potrebne le osnovne funkcije sistema za upravljanje z izvorno kodo. Uporabljen je za shranjevanje in pregled sprememb

v kodi ter za povezavo z sistemom Jenkins [28].

Gradnik, ki predstavlja mapo, v kateri je izvorna koda aplikacije, se imenuje repozitorij. Projekt je potrebno najprej inicalizirati. Če že obstaja Git repozitorij, se ga z "git clone"ukazom, ki se mu doda še pot do repozitorija, klonira v želeno mapo. Če pa je izvorna koda lokalno na disku in Git repozitorij še ne obstaja, se znotraj mape, ki vsebuje datoteke z izvorno kodo, požene ukaz "git init". Ta ukaz doda potrebne datoteke, ki omogočajo delovanje sistema. Ko je repozitorij ustvarjen, se kodo lahko začne spreminjati. Datoteke, pri katerih je prišlo do spremembe, se preveri z ukazom "git status". Ta ukaz vrne seznam datoteke, ki so bile spremenjene. Vse shranjene spremembe se lahko z Git sistemom pregleda, saj sistem vzdržuje zgodovino repozitorija, kamor se shranjujejo te spremembe. Spremembe se lahko pregleda z ukazom "git diff". Pogled sprememb je mogoč za cel repozitorij, če pa je dodana še pot do datoteke, se prikažejo samo spremembe v določeni datoteki. Z ukazom "git addše označi datoteke, ki se bodo dodale v skupek datotek za shranitev. Ukaz "git status"poleg spremenjenih datotek pokaže, katere spremenjene datoteke so bile dodane in katere ne. Ko so datoteke dodane, se jih shrani lokalno v Git repozitorij z ukazom "git commit", ki se mu doda še ime z opcijo -m". Dodano ime služi lažjemu pregledu po Git veji. Privzeto se vse shranjuje na glavno vejo, ki se imenuje "master". Pri več razlogih pride v poštev tudi ustvarjanje drugih vej, ki delujejo paralelno z glavno vejo. Novo vejo se ustvari z ukazom "git checkout", ki ima dodano še opcijo -b"in ime veje. Z uporabo ukaza "git checkout"brez opcije in samo imenom veje se lahko uporabnik premika iz veje na vejo. Za združevanje vej se uporablja ukaz "git mergež dodanim imenom veje, ki združi imenovano vejo skupaj z vejo, kjer se nahaja uporabnik. Pri združevanju lahko pride do konfliktov, ki jih mora uporabnik popraviti, če jih sistem ne popravi sam. Vse opisano se dogaja lokalno, z uporabo ukaza "git push"pa se to naloži na strežnik, kjer se nahaja repozitorij. V nasprotno smer deluje ukaz "git pull", ki s strežnika naloži vse spremenjene datoteke.

Z zgoraj naštetimi ukazi je sistem Git že uporaben za razvijanje srednje

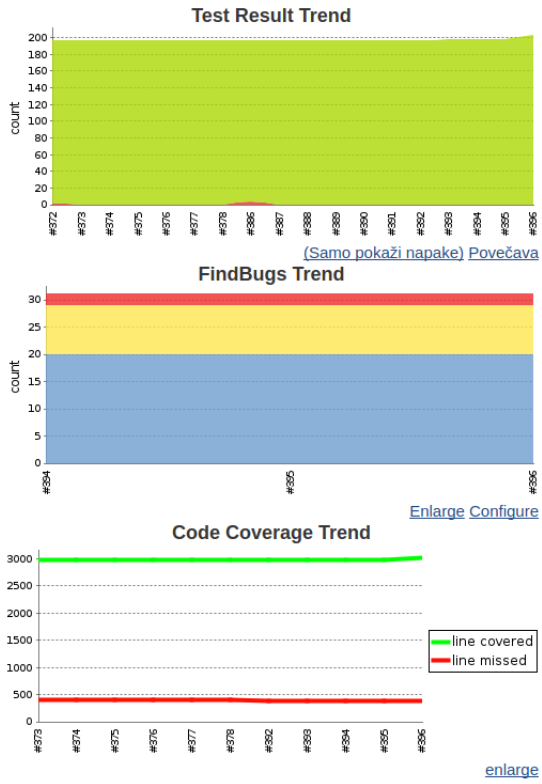
velikih aplikacij. Poleg ukazov so za Git izdelani tudi grafični vmesniki, ki nudijo predvsem boljši pregled. Za izvedbo zahtevnejših operacij so še vedno bolj priporočljivi ukazi.

5.6 Uporaba sistema Jenkins

Sistem Jenkins [28] je uporabljen za avtomatsko izgradnjo aplikacije iz izvorne kode, ki se nahaja v Git repozitoriju. Po izgradnji aplikacije, se izvedejo tudi Unit testi, s pomočjo katerih so narejeni grafi, ki prikazujejo informacije o pokritosti in statični analizi kode.

Za takšno avtomatizacijo je potrebno ustvariti delo v sistemu Jenkins. Delo je sestavljeno iz nastavitev, ki poskrbijo za uporabo potrebnih gradnikov in zaporedje opravljenih korakov. Nastaviti je potrebno Java SDK, da se aplikacija lahko zgradi. Da delo pridobi ustrezno izvorno kodo, se navede pot do Git repozitorija in vejo, na kateri se nahaja koda. Če gre za varovan repozitorij, se določi še uporabnika in njegovo geslo za dostop. Git repozitorij mora vsebovati ustrezno Gradle skripto za grajenje aplikacije, ki vsebuje funkcije za izdelavo poročil o pokritosti kode, najdenih napakah in statični analizi kode. Da se te funkcije izvedejo, je v nastavitvah dela potrebno nastaviti ime Gradle ovojnice, ki se sklicuje na skripto za grajenje. S temi nastavitami sistem Jenkins lahko zgradi aplikacijo. Za poročilo o aplikaciji je potrebno dodati še poti do poročil Unit testov in najdenih napak. Dodatno se lahko nastavi tudi interval za gradnjo aplikacije. Delo se nato shrani v seznam del, ki so izpisana na začetni strani Jenkinsa.

Iz seznama je mogoče razbrati osnovne informacije o delu, kdaj je bilo zadnje uspešno ali neuspešno opravljeno delu in koliko časa je trajalo. Sistem nudi tudi informacije o posamezni gradnji aplikacije, ki so bile nastavljene v nastavitvah.



Slika 5.9: Primer grafov dela sistema Jenkins.

Poglavje 6

Nadaljnji razvoj

6.1 Šifriran video klic

Nadaljnji razvoj aplikacije bi lahko predstavljal glasovni klic, kjer bi se dva uporabnika lahko pogovarjala v živo. Pri tem bi bilo seveda potrebno poskrbeti za varnost. To na prvi pogled ne predstavlja težav, saj obstaja implementirano šifrirano sporočanje prek besedilnih sporočil, vendar to ni tako preprosto. Že pri navadnem video pogovoru pride do zaostanka, odmevov in slabe kvalitete, če pa je temu dodano še šifriranje, se težavnost razvoja video klica še poveča.

Kljub zahtevnosti je implementacija video klica mogoča, saj jo določene aplikacije že uporabljajo. Poleg tega bi bili opisani načini in implementacije uporabni za razvoj video klica. Način shranjevanja in dostop do ključev za šifriranje sta zagotovo uporabna za video klic. Potrebno bi bilo razviti način za ohranjanje klica, saj tudi v tem primeru signal mobilne naprave ni vedno konstanten, nekaj sredstev pa bi se porabilo za optimizacijo klica.

Literatura

[1] Robert C. Martin. *Clean Code*. Pearson Education, 2009.

[2] Niels Ferguson, Bruce Schneier, Tadayoshi Kohno. *Cryptography Engineering*. Wiley, 2010.

[3] Privacy. Dosegljivo: `https://en.wikipedia.org/wiki/Privacy`. [Dostopano 8.1.2018].

[4] Privacy - Types of Privacy. Dosegljivo: `https://www.dlsweb.rmit.edu.au/toolbox/knowmang/content/privacy/types_of_privacy.htm`. [Dostopano 8.1.2018].

[5] Zasebnost. Dosegljivo: `https://sl.wikipedia.org/wiki/Zasebnost`. [Dostopano 8.1.2018].

[6] Kriptografija. Dosegljivo: `http://www.s-sers.mb.edus.si/gradiva/rac/moduli/upravljanje_ik/14_kriptiranje/01_datoteka.html`. [Dostopano 8.1.2018].

[7] Digital Watermarking. Dosegljivo: `https://en.wikipedia.org/wiki/Digital_watermarking`. [Dostopano 8.1.2018].

[8] Symetric-key algorithm. Dosegljivo: `https://en.wikipedia.org/wiki/Symmetric-key_algorithm`. [Dostopano 8.1.2018].

[9] Public-key cryptography. Dosegljivo: `https://en.wikipedia.org/wiki/Public-key_cryptography`. [Dostopano 8.1.2018].

[10] HMAC. Dosegljivo: https://en.wikipedia.org/wiki/Hash-based_message_authentication_code. [Dostopano 8.1.2018].

[11] Android Developers. Dosegljivo: https://developer.android.com/index.html. [Dostopano 8.1.2018].

[12] Android Studio. Dosegljivo: https://developer.android.com/studio/index.html. [Dostopano 8.1.2018].

[13] Android (operating system). Dosegljivo: https://en.wikipedia.org/wiki/Android_(operating_system). [Dostopano 8.1.2018].

[14] Java (programming language). Dosegljivo: https://en.wikipedia.org/wiki/Java_(programming_language). [Dostopano 8.1.2018].

[15] SpongyCastle. Dosegljivo: http://rtyley.github.io/spongycastle/. [Dostopano 8.1.2018].

[16] XMPP. Dosegljivo: https://xmpp.org/. [Dostopano 8.1.2018].

[17] Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. Dosegljivo: https://xmpp.org/rfcs/rfc6121.html. [Dostopano 8.1.2018].

[18] XEP-0045: Multi-User Chat. Dosegljivo: https://xmpp.org/extensions/xep-0045.html. [Dostopano 8.1.2018].

[19] Ignite Realtime: Smack API. Dosegljivo: https://www.igniterealtime.org/projects/smack/. [Dostopano 8.1.2018].

[20] Multi-User Chat Subscriptions. Dosegljivo: https://docs.ejabberd.im/developer/xmpp-clients-bots/proposed-extensions/muc-sub/. [Dostopano 8.1.2018].

[21] Gradle. Dosegljivo: https://gradle.org/. [Dostopano 8.1.2018].

[22] XML. Dosegljivo: https://www.w3.org/XML/. [Dostopano 8.1.2018].

[23] Git. Dosegljivo: https://en.wikipedia.org/wiki/Git. [Dostopano 8.1.2018].

[24] JSON. Dosegljivo: https://www.json.org/. [Dostopano 8.1.2018].

[25] Unit testing. Dosegljivo: https://en.wikipedia.org/wiki/Unit_testing. [Dostopano 8.1.2018].

[26] HTTP methods. Dosegljivo: https://www.w3schools.com/tags/ref_httpmethods.asp. [Dostopano 8.1.2018].

[27] Hypertext transfer Protocol. Dosegljivo: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. [Dostopano 8.1.2018].

[28] Jenkins. Dosegljivo: https://jenkins-ci.org/. [Dostopano 8.1.2018].

[29] Code coverage. Dosegljivo: https://en.wikipedia.org/wiki/Code_coverage. [Dostopano 8.1.2018].

[30] Static program analysis. Dosegljivo: https://en.wikipedia.org/wiki/Static_program_analysis. [Dostopano 8.1.2018].